1987-03

Image texture generation using autoregressive integrated moving average (ARIMA)--mode

Rathmanner, Steven Clifford

NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

IMAGE TEXTURE GENERATION USING AUTOREGRESSIVE INTEGRATED MOVING AVERAGE (ARIMA) MODELS

by

Steven Clifford Rathmanner

March 1987

Thesis Advisor: Charles W. Therrien

Approved for public release; distribution is unlimited

T233631

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b RESTRICTIVE MARKINGS
2a SECURITY CLASSIFICATION AUTHORITY	3 DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution is unlimited
2b DECLASSIFICATION / DOWNGRADING SCHEDULE	
4 PERFORMING ORGANIZATION REPORT NUMBER(S)	5 MONITORING ORGANIZATION REPORT NUMBER(S)
6a NAME OF PERFORMING ORGANIZATION Naval Postgraduate School; 6b OFFICE SYMBOL (If applicable) Code 62	7a NAME OF MONITORING ORGANIZATION Naval Postgraduate School
6c ADDRESS (City, State, and ZIP Code) Monterey, California 93943-5000	7b ADDRESS (City, State, and ZIP Code) Monterey, California 93943-5000
8a NAME OF FUNDING/SPONSORING ORGANIZATION; 8b OFFICE SYMBOL (If applicable)	9 PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c ADDRESS (City, State, and ZIP Code)	10 SOURCE OF FUNDING NUMBERS: PROGRAM ELEMENT NO; PROJECT NO; TASK NO; WORK UNIT ACCESSION NO

11 TITLE (Include Security Classification)
IMAGE TEXTURE GENERATION USING AUTOREGRESSIVE INTEGRATED MOVING AVERAGE (ARIMA) MODELS

12 PERSONAL AUTHOR(S)
Rathmanner, Steven C.

13a TYPE OF REPORT	13b TIME COVERED	14 DATE OF REPORT (Year, Month, Day)	15 PAGE COUNT
Master's Thesis	FROM ______ TO ______	1987, March	132

16 SUPPLEMENTARY NOTATION

17 COSATI CODES			18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	Autoregressive Integrated Moving Average (ARIMA Stationary; Separable

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

This thesis involves investigation of linear filtering models as a means of generating texture in images. Various autoregressive filter models are used to generate various textures, and the results are analyzed to determine relationships between filter parameters and texture characteristics. A two-dimensional counterpart to the autoregressive integrated moving average (ARIMA) model from one-dimensional time series analysis theory is developed and tested for texture modeling applications. All these models are driven by white noise, and to the extent that real images can be reproduced this way, advantages in image texture transmission could be realized. Results of this work indicate that the purely autoregressive models work well for some types of image textures, but that for the textures studied the ARIMA model is not particularly suitable.

20 DISTRIBUTION/AVAILABILITY OF ABSTRACT [✓ UNCLASSIFIED/UNLIMITED] ☐ SAME AS RPT ☐ DTIC USERS	21 ABSTRACT SECURITY CLASSIFICATION Unclassified
22a NAME OF RESPONSIBLE INDIVIDUAL Prof. C.W. Therrien	22b TELEPHONE (Include Area Code) (408) 646-3347; 22c OFFICE SYMBOL Code 62Ti

DD FORM 1473, 84 MAR   83 APR edition may be used until exhausted
All other editions are obsolete

SECURITY CLASSIFICATION OF THIS PAGE
UNCLASSIFIED

Image Texture Generation Using Autoregressive Integrated Moving Average (ARIMA) Models

by

Steven Clifford Rathmanner
Lieutenant, United States Navy
B.S., Florida State University, 1979

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
March 1987

ABSTRACT

This thesis involves investigation of linear filtering models as a means of generating texture in images. Various autoregressive filter models are used to generate various textures, and the results are analyzed to determine relationships between filter parameters and texture characteristics. A two-dimensional counterpart to the autoregressive integrated moving average (ARIMA) model from one-dimensional time series analysis theory is developed and tested for texture modeling applications. All these models are driven by white noise, and to the extent that real images can be reproduced this way, advantages in image texture transmission could be realized. Results of this work indicate that the purely autoregressive models work well for some types of image textures, but that for the textures studied the ARIMA model is not particularly suitable.

DISCLAIMER

Some terms used in this thesis are registered trademarks of commercial products. Rather than attempt to cite each occurrence of a trademark, all trademarks appearing in this thesis are listed below following the name of the firm holding the trademark:

1. COMTAL, Altadena, California

   Vision One/20

2. Digital Equipment Corporation, Maynard, Massachusetts

   VAX/UNIX   VAX/VMS

3. International Business Machines Corporation, Armonk, New York

   IBM System/370 3033

TABLE OF CONTENTS

I. INTRODUCTION

The purpose of this thesis is to investigate the types and quantity of image textures generated using a two-dimensional (2-D) extension of the Autoregressive Integrated Moving Average (ARIMA) model. For the one-dimensional (e.g., time series) case, the theories and formulas describing this model are outlined in Box and Jenkins [Ref. 1:pp. 85-103]. The 1-D ARIMA model is useful when the time series to be modeled is not stationary but exhibits some homogeneity in the sense that, except for statistical differences between parts of the time series, these different parts of the process behave similarly. In these cases some suitable difference of the process may be stationary, and hence may be accurately modeled by an Autoregressive Moving Average (ARMA) or a purely Autoregressive (AR) model. The resulting stationary time series (generated by an appropriate ARMA or AR filter with white noise input) is applied to an integra-tion or summation filter (the inverse of the difference operation) to generate the original nonstationary time series. [Ref. 1:p. 85] Figure 1-1 shows a block diagram of this process.

This work attempts to extend these concepts to two-dimensional signal processing. In order to simplify the model, the moving average (MA) portion of it will be

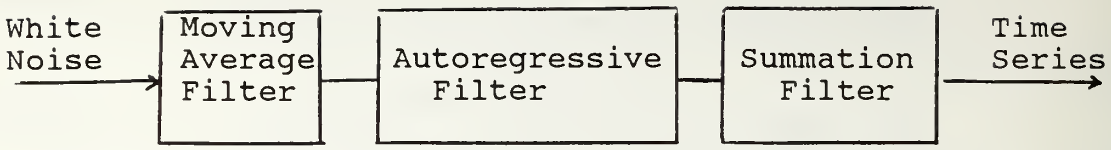


Figure 1-1 Block Diagram for the Autoregressive Integrated Moving Average Model

eliminated (i.e., no zeros in the filter Z transform) so that only purely AR models will be considered for stationary image generation. The procedures for modeling image textures using AR models with white noise input are well established [Ref. 2:pp. 454-456]. However, a suitable two-dimensional difference operation and its inverse must be found to implement the concepts outlined above.

The research is divided into four areas:

1) Investigation of the various types of image textures generated using AR models where filter coefficients and size are determined a) arbitrarily, b) using a two-pole separable model, and c) using a four-pole separable model. Separability refers to the fact that the Z transform of the AR filter can be factored into components representing each dimension or direction of the image.

2) Selection of a difference operator and a realizable inverse (integration or summation) filter.

3) Application of the above autoregressively generated images to the summation filter, and evaluation of these results.

4) Attempted reproduction of actual images textures using AR models whose coefficients are determined using the statistics of the image, and comparison of these results to those obtained by a) applying the difference operator to the real image, b) finding the coefficients of the AR model that reproduce the difference image, and c) applying the difference image to the summation filter. This comparison was intended

to discover what improvements, if any, may be realized using the ARIMA model vice a purely ARMA (or AR) model.

The remainder of the thesis is organized as follows. Chapter II contains methods and results of investigating various autoregressive image models. Chapter III deals with the formulation, development, and testing of the two-dimensional summation filter. Chapter IV contains the results of applying various AR-generated images to the summation filter and addresses the application of the ARIMA model to real image textures. Chapter V outlines conclusions on the results and applicability of the ARIMA model. Although the ARIMA modeling was not highly successful in reproducing the textures studied here, plausible reasons are given for their failure and conjectures are made about those circumstances where the model would be more successful. Appendix A provides information on the computer algorithms used to implement the equations governing the above processes. Appendices B through G contain derivations of spectral and autocorrelation equations, and the corresponding graphical results, governing the AR processes in Chapter II. Appendices H and I contain graphical results associated with the inverse filter development in Chapter III.

Image data were generated using computer programs written in FORTRAN, compiled using Version 4.5 under the VAX/VMS Version 4.4 operating system. The images were displayed on the COMTAL Vision One/20. The gray level intensity range of pixel values is 0 (darkest) to 255 (lightest), so the image data generated had to be scaled to that range for display (see Appendix A).

II. THE AUTOREGRESSIVE IMAGE MODEL

A two-dimensional signal (such as an image texture) can be modeled using a two-dimensional AR model with white noise input. The governing equations in the spatial domain are of the following form [Ref. 3:pp. 325-326]:

$$y(n,m) = -\sum_{\substack{i=0 \\ (i,j)\neq(0,0)}}^{P-1} \sum_{j=0}^{Q-1} a_{ij} y(n-i,m-j) + w(n,m) \qquad (2.1)$$

where y(n,m) is a signal representing the generated image texture at pixel location (n,m), a_{ij} is the filter coefficient matrix, and w(n,m) is a two-dimensional white noise signal. The system function corresponding to the filter of Eq. (2.1) is given by

$$Y(z_1,z_2) = \frac{1}{1+a_{10}z_1^{-1}+a_{01}z_2^{-1}+a_{11}z_1^{-1}z_2^{-1}+\ldots+a_{P-1Q-1}z_1^{-(P-1)}z_2^{-(Q-1)}} \cdot W(z_1,z_2) \qquad (2.2)$$

where z_1 and z_2 are the Z transform variables corresponding to spatial coordinates n and m. Ideal white noise has an autocorrelation function that is an impulse and a flat (constant) power spectrum with magnitude corresponding to the variance of the white noise process [Ref. 4:pp. 22-26]. Therefore determination of the filter coefficients a_{ij} will define the generated image process. Procedures will be outlined later to estimate the coefficients from real image data. At this point analytical methods will be used to select these coefficients and the resulting images will be studied. Figure 2-1 shows an example of a white noise input image.

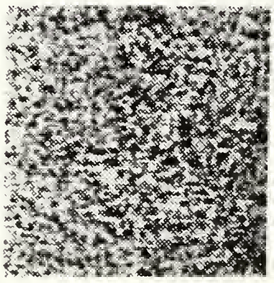
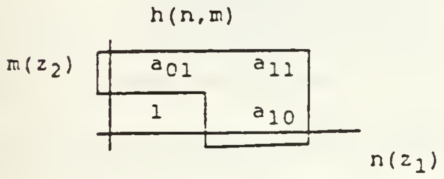


Figure 2-1 White Noise Image

A. ARBITRARILY SELECTED FILTER COEFFICIENTS; P = 2, Q = 2

In order to get an initial idea of what types of images might be generated using a 2×2 AR filter with white noise input, filter coefficients were at first selected arbitrarily, but subject to a stability constraint. The primary constraint on coefficient selection is that of filter stability. Using the DeCarlo-Strintzis Theorem dealing with multidimensional filter stability [Ref. 3:pp. 197-198], alternately setting $z_1 = 1$ and $z_2 = 1$ and determining the location of the pole in the remaining dimension will indicate whether or not the filter is stable. If the magnitude of the pole in the remaining dimension is less than 1, the filter is stable. Even with this condition, however, there are an infinite number of possible filter coefficient combinations. The additional constraint of $a_{10} = a_{01}$ can be used, and comparisons of results using various values of a_{11} can be made.

Figure 2-2 shows the form and directionality convention used for the autoregressive filter, along with the corresponding difference equation and its Z transform.



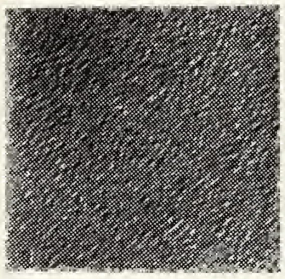
$$y(n,m) = - \sum_{\substack{i=0 \\ (i,j)\neq(0,0)}}^{1} \sum_{j=0}^{1} a_{ij} \cdot y(n-i,m-j) + w(n,m)$$

$$Y(z_1,z_2) = H(z_1,z_2) \cdot W(z_1,z_2) = \frac{1}{1+a_{10}z_1^{-1}+a_{01}z_2^{-1}+a_{11}z_1^{-1}z_2^{-1}} \cdot W(z_1,z_2)$$

Figure 2-2 Autoregressive Filter Impulse Response, Difference Equation, and Z Transform

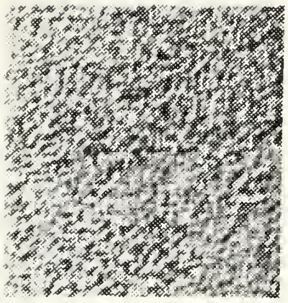
Although it is difficult to make precise predictions in two dimensions, one can expect that the sign and magnitude of a_{10} or a_{01} would influence the correlation between pixels in the corresponding directions. For example, a positive value for a_{10} might be expected to yield an image with substantial variation in the n direction (low correlation), particularly if the magnitude of a_{10} is near 1. A negative value for a_{01} with magnitude near 1 might yield an image with lower variations in pixel intensity (high correlation) in the m direction. Since the filter is not necessarily separable (i.e., the denominator of $H(z_1,z_2)$ cannot be factored into the form $D(z_1)\cdot D(z_2)$), conclusions drawn from this line of reasoning may not be completely correct.

Initial attempts at generating images with arbitrarily selected coefficients yielded rather uninteresting results having very little contrast or discernible pattern. Continued experimentation with combinations where $a_{10} = a_{01} < 0$ eventually yielded more interesting image textures. Figures 2-3 and 2-4 show the results of using the constraint $a_{10} = a_{01} = -0.35$ and various values of a_{11} for the filter coefficients. For positive values of a_{11}, the images are rather "grainy," with higher magnitudes yielding a somewhat "finer" graininess. There are also some overall intensity differences observed. For the negative values of a_{11}, the results are much more interesting. As the magnitude increases, there is a gradually more noticeable upper left to lower right orientation of the image texture, and the variations from lower left to upper right become smoother as well. Using $a_{10} = a_{01} = -0.38$ and $a_{11} = -0.24$



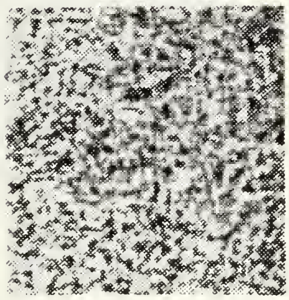
$a_{10} = a_{01} = -0.35$

$a_{11} = 0.8$



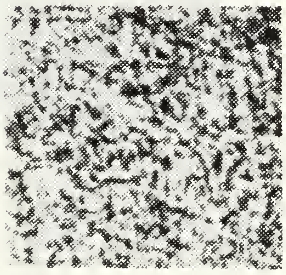
$a_{10} = a_{01} = -0.35$

$a_{11} = 0.5$



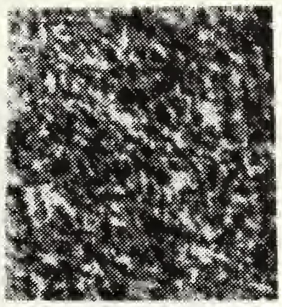
$a_{10} = a_{01} = -0.35$

$a_{11} = 0.2$



$a_{10} = a_{01} = -0.35$

$a_{11} = 0.0$



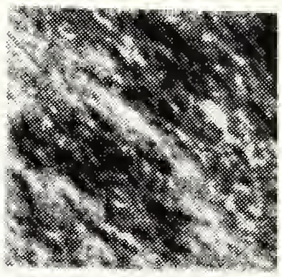
$a_{11} = a_{01} = -0.35$

$a_{11} = -0.1$

$a_{10} = a_{01} = -0.35$

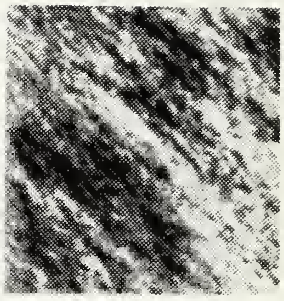
$a_{11} = -0.2$

Figure 2-3 Images Generated Using Arbitrarily Selected Filter Coefficients



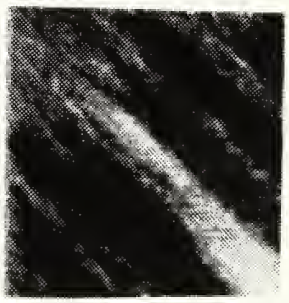
$a_{10} = a_{01} = -0.35$

$a_{11} = -0.25$



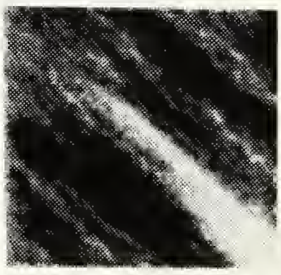
$a_{10} = a_{01} = -0.35$

$a_{11} = -0.27$



$a_{10} = a_{01} = -0.35$

$a_{11} = -0.3$



$a_{10} = a_{01} = -0.38$

$a_{11} = -0.24$

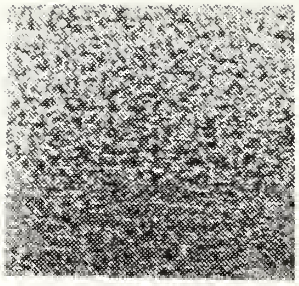
Figure 2-4 Images Generated Using Arbitrarily Selected Filter Coefficients

yields minor variations in texture pattern and overall image intensity when compared to the previous case.

Using $a_{10} = a_{01} = 0.35$ and varying a_{11} from 0.0 to 0.8 (Figure 2-5), the images obtained deviate very little from the mean intensity value, and possess minor differences in graininess. With these positive coefficients some negative correlation might be expected, and the fact that these images are "grainy" indicates the existence of some negative correlation or high spatial frequency characteristics. On initial examination, however, the low contrast of the generated images tends to obscure the observed graininess.

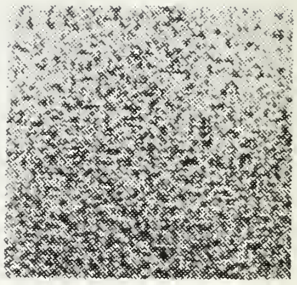
B. TWO POLE, SEPARABLE AUTOREGRESSIVE MODEL

In general, it is difficult to relate the nature or properties of a two-dimensional filter to the precise nature of an image texture that may be generated when white noise is applied to that filter. In order to simplify the effort and to obtain a better understanding of the problem, the case where the filter (and resulting image texture) are separable is considered. For the two pole separable case considered here, the filter transfer function can be factored into expressions in z_1 alone and z_2 alone. The expressions in z_1 and z_2 each have one pole on the real axis in their respective Z domains. Figure 2-6 illustrates the filter structure, the corresponding difference equation, and its Z transform.



$a_{10} = a_{01} = 0.35$

$a_{11} = 0.0$



$a_{10} = a_{01} = 0.35$

$a_{11} = 0.2$

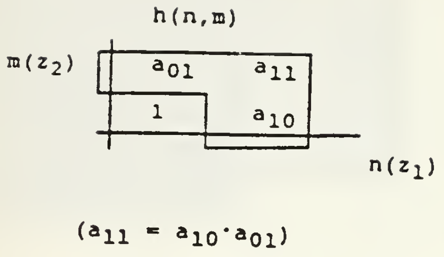
$a_{10} = a_{01} = 0.35$

$a_{11} = 0.5$

$a_{10} = a_{01} = 0.35$

$a_{11} = 0.8$

Figure 2-5 Images Generated Using Arbitrarily Selected Filter Coefficients



$$y(n,m) = -\sum_{i=0}^{1}\sum_{\substack{j=0\\(i,j)\neq(0,0)}}^{1} a_{ij}\cdot y(n-i,m-j) + w(n,m)$$

$$Y(z_1,z_2) = H(z_1,z_2)\cdot W(z_1,z_2) = \frac{1}{1+a_{10}z_1^{-1}}\cdot\frac{1}{1+a_{01}z_2^{-1}}\cdot W(z_1,z_2)$$

$$= \frac{1}{1+a_{10}z_1^{-1}+a_{01}z_2^{-1}+a_{10}\cdot a_{01}z_1^{-1}z_2^{-1}}\cdot W(z_1,z_2)$$

Figure 2-6 Autoregressive Filter Form, Difference Equation, and Z Transform

Here it is relatively easy to relate stability of the filter to the location of the poles in the z_1 and z_2 planes. Since the quarter plane filter is separable and the components are causal, one-dimensional filter stability theory can be used to state that the poles in each plane must have magnitude less than 1 to ensure filter stability. Figures 2-7 through 2-9 show images resulting from this model for various values of a_{10} and a_{01}. Note that the sign convention used for the filter difference equation and Z transform results in poles on the negative side of the real axis for positive values of a_{10} or a_{01}, and vice versa.

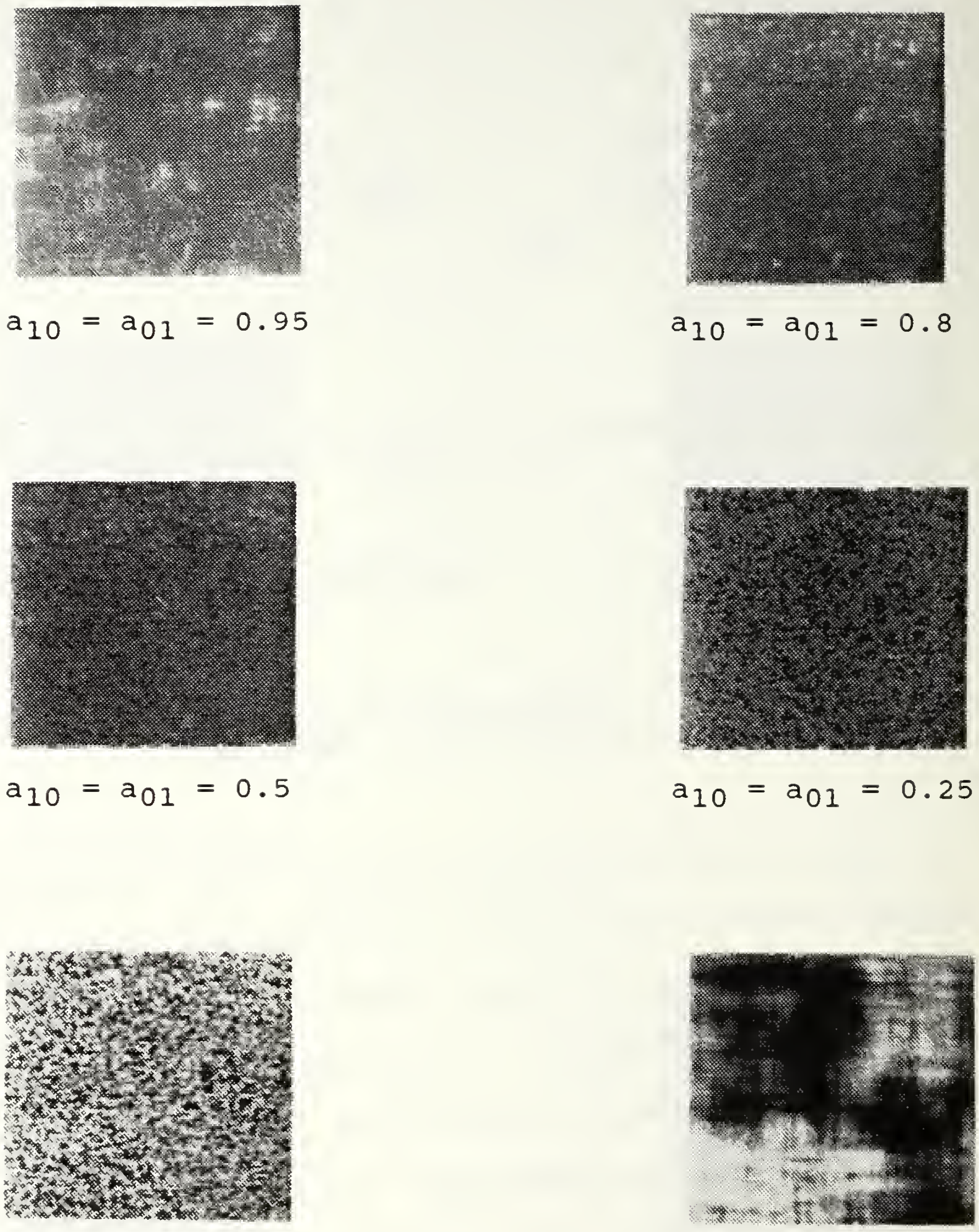
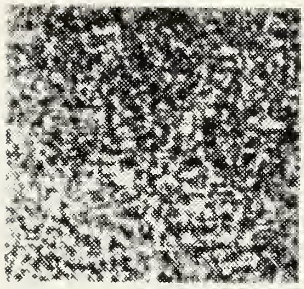
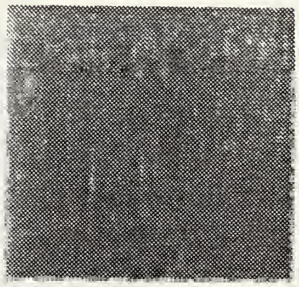


Figure 2-7 Images Generated Using a Two-Pole Autoregressive Model

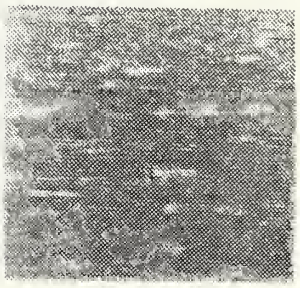


$a_{10} = a_{01} = -0.25$

$a_{10} = 0.95 \quad a_{01} = 0.5$



$a_{10} = 0.5 \quad a_{01} = 0.95$

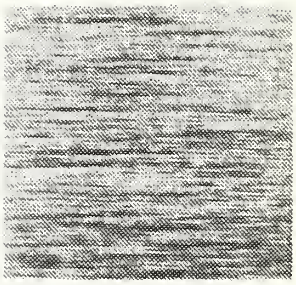


$a_{10} = 0.95 \quad a_{01} = 0.75$

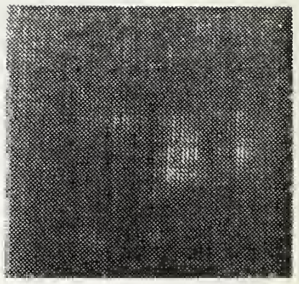
$a_{10} = -0.95 \quad a_{01} = 0.25$

$a_{10} = -0.95 \quad a_{01} = -0.25$

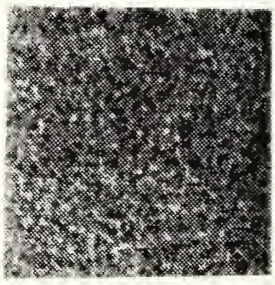
Figure 2-8 Images Generated Using a Two-Pole Autoregressive Model



$a_{10} = -0.95 \quad a_{01} = 0.75$



$a_{10} = 0.95 \quad a_{01} = -0.95$



$a_{10} = 0.25 \quad a_{01} = -0.25$

Figure 2-9 Images Generated Using a Two-Pole Autoregressive Model

Careful comparison of the images resulting from various combinations of a_{10} and a_{01} leads to the following observations:

1) When poles are located in the same place in the z_1 and z_2 planes on the negative side of the real axis, magnitudes near 1 yield a fine graininess with patchy areas and low overall contrast. As the magnitude of the pole decreases, the graininess becomes more coarse and the result is more like the original white noise input. No directional quality in the image pattern is observed.

2) When poles are located in the same place on the positive side of the z_1 and z_2 real axes, a somewhat different result is observed. For magnitudes near 1, an image of patchy light and dark areas results, with differing amounts of correlation between pixels in different areas. Slightly discernible "lines" in both the horizontal and vertical directions are also observed. For lower pole magnitudes on the positive side of the real axis, the decreased effect of the filter on the white noise input is again observed. This result is more like the white noise and has more contrast than the corresponding result using poles on the negative side of the real axis.

3) For pole placements in the z_1 and z_2 planes which are on the negative side of the real axis and are of unequal magnitude, the results have a very fine graininess and low contrast. Some slight directionality is observable in the image patterns, with lower frequency variations evident in the direction corresponding to the pole with smaller magnitude.

4) For pole placements in the z_1 and z_2 planes which are on the positive side of the real axis and are of unequal magnitude, much more directionality and variation is observable in the image pattern.

5) As the poles are placed on opposite sides of the real axis and are separated by a greater distance, directionality becomes more evident (with higher frequency variations in the direction of the more negative pole). As the pole separation becomes greater and as the pole magnitudes become closer to 1, smoother sinusoidal variations are evident.

6) When the pole magnitudes are equal and have opposite sign, the image generated using pole magnitudes close

to 1 exhibits high frequency sinusoidal variations in the direction of the negative pole. The image generated with the lower magnitude poles, as would be expected from the above results, resembled the unfiltered white noise.

In order to explain these image patterns analytically, analysis of the power spectrum and autocorrelation function of this process is useful. Since this model is separable, the analysis can be conducted in each direction separately. The power spectrum is defined by [Ref. 4:pp. 24-34]:

$$S_y(\omega) = \sigma^2 |H(e^{j\omega})|^2 = \sigma^2 H(e^{j\omega}) H(e^{-j\omega}) \qquad (2.3a)$$

where

$$H(e^{j\omega}) = H(z)|_{z=e^{j\omega}} \qquad (2.3b)$$

and for this case

$$H(z) = \frac{1}{1 + \alpha z^{-1}} \qquad (2.3c)$$

Here σ^2 is the magnitude of the white noise power spectrum. We can assume that $\sigma^2 = 1$ with no loss of generality of the results, since σ^2 does not affect the shape of the frequency response.

The autocorrelation function is related to the filter transfer function through the equations [Ref. 5:pp. 391-395]:

$$h(n) = Z^{-1}[H(z)] \tag{2.4a}$$

$$y(n) = h(n)*w(n) \tag{2.4b}$$

$$R_y(\ell) = \sigma^2 \sum_{n=-\infty}^{\infty} h(n)\cdot h(n-\ell) \tag{2.4c}$$

Specific forms of the power spectrum and autocorrelation function are given in Appendix B. Since $R_y(\ell) = R_y(-\ell)$ [Ref. 5:p. 388], calculating the expression for $R_y(\ell)$, $\ell < 0$ is not necessary. From Appendix B, the results are:

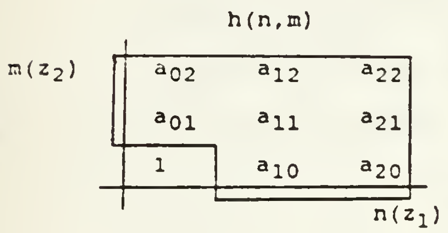
$$S_y(\omega) = \frac{1}{1 + 2\alpha\cos(\omega) + \alpha^2} \tag{2.5}$$

$$R_y(\ell) = \frac{(-\alpha)^\ell}{1 - \alpha^2} \qquad \ell \geq 0 \tag{2.6}$$

Appendix C shows the results of these equations graphically for various values of α. The relationship between the power spectra and their corresponding autocorrelation functions conforms to the expected results from theory (i.e., low frequency spectrum with smooth autocorrelation function, and high frequency power spectrum with rapidly varying autocorrelation function) [Ref. 6:pp. 139-142]. The plots in Appendix C also demonstrate that: 1) For poles on the positive side of the real axis in the Z plane low frequencies predominate and for poles on the negative side of the real axis high frequencies predominate, and 2) Lower magnitudes of α result in a broader power spectrum and a wider range of frequencies of significant magnitude. Both of these observations agree with the image results. For images generated using a more negative pole in a given direction, fine, high frequency graininess is observed in that direction (though the low contrast or low variation about the mean intensity may tend to make this effect less noticeable). When a more positive pole is used, lower frequency variations are more evident in the corresponding direction. As lower magnitudes are used for α in a given direction, more random variations (indicative of a wider range of significant frequency components) are observed in that direction. The form of the autocorrelation function for these cases approaches the autocorrelation function for white noise, i.e., an impulse. Negative poles should yield high frequencies since the negative side of the real axis in the Z plane represents a digital spatial frequency of π, while positive poles in the Z plane correspond to a digital spatial frequency of zero. Note that even when the poles are placed such that a spatial frequency of zero should predominate, there are some low frequency random variations in the resulting images. Since the power spectra of the positive poles all contain some non-zero frequency components (they are not perfect impulses at zero), this characteristic is expected.

C. FOUR POLE, SEPARABLE AUTOREGRESSIVE MODEL

For the cases considered in this section, H(z) again can be factored into expressions in z_1 and z_2. However here each factor is a 2nd degree polynomial with two poles in the denominator. Figure 2-10 illustrates the filter structure, the applicable difference equation, and the corresponding Z transform.



$$y(n,m) = -\sum_{\substack{i=0 \\ (i,j)\neq(0,0)}}^{2} \sum_{j=0}^{2} a_{ij} \cdot y(n-i,m-j) + w(n,m)$$

$$Y(z_1,z_2) = H(z_1,z_2)\cdot W(z_1,z_2) = \frac{1}{1+a_{10}z_1^{-1}+a_{20}z_1^{-2}} \cdot \frac{1}{1+a_{01}z_2^{-1}+a_{02}z_2^{-2}} \cdot W(z_1,z_2)$$

$$= \frac{1}{1+a_{10}z^{-1}+a_{20}z_1^{-2}+a_{01}z_2^{-1}+a_{02}z_2^{-2}+a_{11}z_1^{-1}z_2^{-1}+a_{21}z_1^{-2}z_2^{-1}+a_{12}z_1^{-1}z_2^{-2}+a_{22}z_1^{-2}z_2^{-2}} \cdot W(z_1,z_2)$$

where

$$a_{11} = a_{10}\cdot a_{01};\quad a_{21} = a_{10}\cdot a_{02};\quad a_{12} = a_{02}\cdot a_{10};\quad a_{22} = a_{20}\cdot a_{02}$$

Figure 2-10 Autoregressive Filter Form, Difference Equation, and Z Transform

Since all of the a_{ij} coefficients are real, the poles must 1) both be on the real axis, or 2) occur in complex conjugate pairs in the z_1 and z_2 planes. Again, pole magnitudes must be less than 1 to ensure filter stability. We will assume here that the poles in each of the factors have equal magnitudes and opposite (or 0 or $\pm\pi$) phase. Letting

α_1 = magnitude of poles in the z_1 plane

θ_1 = pole angle (phase) in the z_1 plane

α_2 = magnitude of poles in the z_2 plane

θ_2 = pole angle (phase) in the z_2 plane

and using Euler's relation, the denominators of $H(z_1)$ and $H(z_2)$ can be expressed as follows:

$$1+a_{11}z_1^{-1}+a_{20}z_1^{-2} = (1-\alpha_1 e^{j\theta_1} z_1^{-1})(1-\alpha_1 e^{-j\theta_1} z_1^{-1}) = 1-2\alpha_1\cos(\theta_1)\cdot z_1^{-1}+\alpha_1^2 z_1^{-2}$$

$$1+a_{01}z_2^{-1}+a_{02}z_2^{-2} = (1-\alpha_2 e^{j\theta_2} z_2^{-1})(1-\alpha_2 e^{-j\theta_2} z_2^{-1}) = 1-2\alpha_2\cos(\theta_2) z_2^{-1}+\alpha_2^2 z_2^{-2}$$

Hence:

$a_{10} = -2\alpha_1\cdot\cos(\theta_1)$ $\qquad$ $a_{01} = -2\alpha_2\cdot\cos(\theta_2)$

$a_{20} = \alpha_1^2$ $\qquad$ $a_{02} = \alpha_2^2$

This gives a relationship between the pole magnitude and angle in the Z domain and the filter coefficients in the spatial domain.

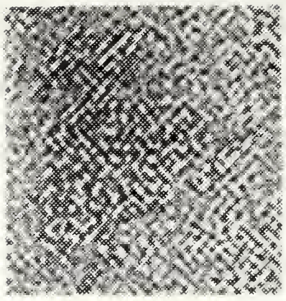
1. Complex Conjugate Poles

Figures 2-11 through 2-13 illustrate images generated using this model for various complex conjugate pole combinations in the z_1 and z_2 directions. In comparing each of these image textures in terms of the relative effect of pole positioning in each direction, the following observations can be made:

1) For images generated using poles of equal magnitude and angle in both directions, graininess with no directionality to the pattern resulted. Higher pole angles yielded finer (higher frequency) graininess and less contrast. Lower magnitude poles yielded a more random and less structured graininess pattern at the same pole angle.

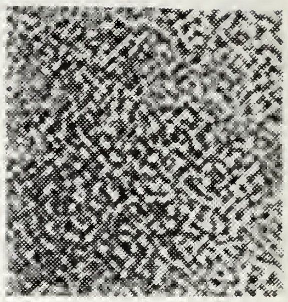
2) Using a pole angle of zero (pole on positive real axis) in one direction and a pole of some non-zero angle in the other direction yielded images with highly directional sinusoidal patterns. The direction corresponding to the pole on the real axis was not totally devoid of variation, but variations were slow, i.e., of very low frequency. The spatial frequency of the sinusoidal pattern can be increased by increasing the pole angle. Large magnitude, high pole angle combinations yielded much cleaner and more structured textures than low magnitude, low pole angle combinations. Lower magnitude, high pole angle combinations yielded less structured textures where directionality was evident but the sinusoidal pattern was obscured. Low magnitude, low pole angle combinations yielded very random, non-structured textures of relatively high contrast.

3) Using poles in the z_1 and z_2 planes with the same magnitude but different pole angles resulted in some directionality if there was a sufficiently large magnitude and difference in the pole angles. As observed earlier, the direction with the higher pole angle had the higher spatial frequency. Large pole magnitudes (close to 1) resulted in more structured but rather low-contrast images (the low contrast seemed to obscure the high frequency nature of the pattern somewhat). Pole angles in z_1 and z_2 that were of close value made it difficult to detect the higher frequency (higher pole angle) direction. Low



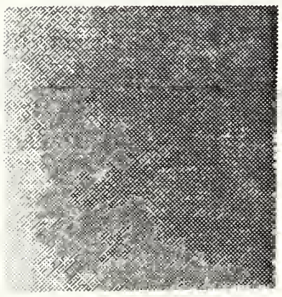
z_1 $0.8e^{\pm j\frac{\pi}{3}}$

z_2 $0.8e^{\pm j\frac{\pi}{3}}$



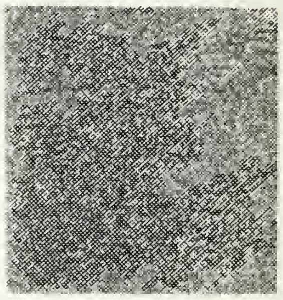
z_1 $0.7e^{\pm j\frac{\pi}{3}}$

z_2 $0.7e^{\pm j\frac{\pi}{3}}$



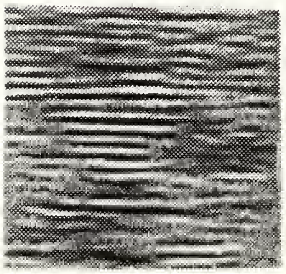
z_1 $0.9e^{\pm j\frac{2\pi}{3}}$

z_2 $0.9e^{\pm j\frac{2\pi}{3}}$



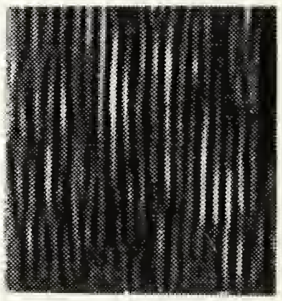
z_1 $0.6e^{\pm j\frac{2\pi}{3}}$

z_2 $0.6e^{\pm j\frac{2\pi}{3}}$



z_1 $0.9e^{\pm j0}$

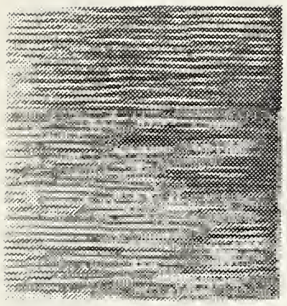
z_2 $0.9e^{\pm j\frac{\pi}{3}}$



z_1 $0.9e^{\pm j\frac{\pi}{3}}$

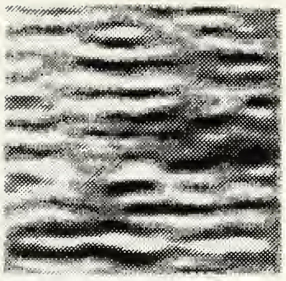
z_2 $0.9e^{\pm j0}$

Figure 2-11 Images Generated Using a Four Pole Autoregressive Model (Poles Listed Below Image)



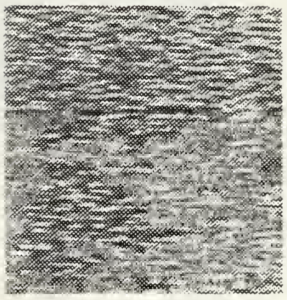
z_1 $0.9e^{\pm j0}$

z_2 $0.9e^{\pm j\frac{2\pi}{3}}$



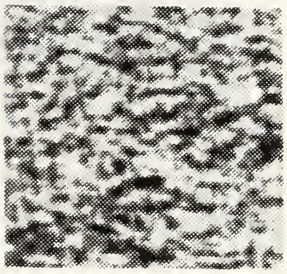
z_1 $0.9e^{\pm j0}$

z_2 $0.9e^{\pm j\frac{\pi}{6}}$



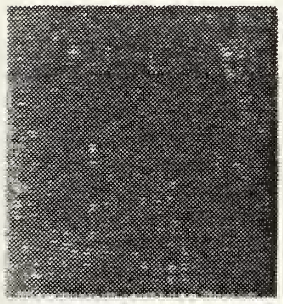
z_1 $0.6e^{\pm j0}$

z_2 $0.6e^{\pm j\frac{2\pi}{3}}$



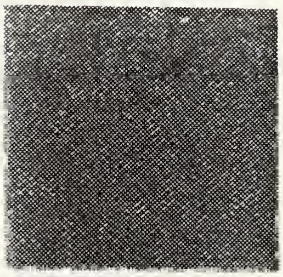
z_1 $0.6e^{\pm j0}$

z_2 $0.6e^{\pm j\frac{\pi}{6}}$



z_1 $0.9e^{\pm j\frac{\pi}{6}}$

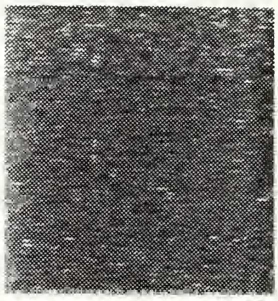
z_2 $0.9e^{\pm j\frac{5\pi}{6}}$



z_1 $0.9e^{\pm j\frac{5\pi}{12}}$

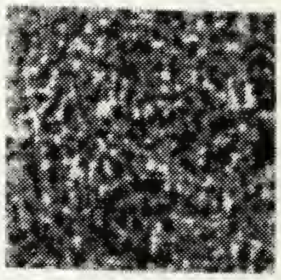
z_2 $0.9e^{\pm j\frac{7\pi}{12}}$

Figure 2-12 Images Generated Using a Four Pole Autoregressive Model (Poles Listed Below Image)



$z_1 \quad 0.6e^{\pm j\frac{\pi}{6}}$

$z_2 \quad 0.6e^{\pm j\frac{5\pi}{6}}$

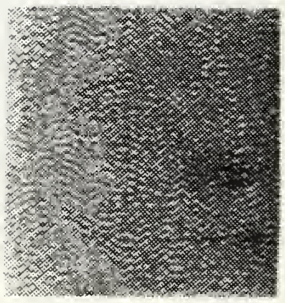


$z_1 \quad 0.5e^{\pm j\frac{\pi}{6}}$

$z_2 \quad 0.5e^{\pm j\frac{\pi}{10}}$

$z_1 \quad 0.9e^{\pm j\frac{\pi}{6}}$

$z_2 \quad 0.8e^{\pm j\frac{\pi}{2}}$



$z_1 \quad 0.6e^{\pm j\frac{5\pi}{12}}$

$z_2 \quad 0.9e^{\pm j\frac{7\pi}{12}}$

Figure 2-13 Images Generated Using a Four Pole Autoregressive Model (Poles Listed Below Image)

magnitudes basically negated the pole angle effects and yielded a very random, unstructured, high contrast texture.

4) Where pole magnitudes were close and pole angles were different in the z_1 and z_2 planes, some directionality in the texture was observed. Again, the high pole angle direction yielded the highest frequency. When the pole angles in both Z domains had similar values and the magnitudes of the poles differed, graininess with little or no discernible directionality or structure resulted.

These observations are consistent with classical pole-zero frequency response analysis [Ref. 7:pp. 323-331]. There is a direct relationship between pole angle and spatial frequency in a given direction, and the magnitude of the poles affects the amount of structure and definition of the sinusoidal pattern of a given frequency in a given direction. Higher magnitude poles result in a narrower bandwidth of the filter and yield more structure and sinusoidal pattern definition. Low pole magnitudes give the filter wider bandwidth and yield images with less structure and definition and more randomness in a given direction. While directional dependencies are evident given pole magnitude and angle in a given direction, it does not appear that a pattern in one direction is totally independent of a pattern in the other direction. This would be expected, even though the model is separable, due to the cross terms in the filter structure.

Filter power spectrum and autocorrelation analysis can be conducted in this case, as in the case of the two pole model. The derivations for $S_y(\omega)$ and $R_y(\ell)$ are somewhat more involved, and are given in Appendix D. The resulting expressions for $S_y(\omega)$ and $R_y(\ell)$ from Appendix D are:

$$S_y(\omega) = \frac{1}{1+\alpha^4+2(\alpha^2-2[\alpha^3-\alpha]\cos(\theta)\cos(\omega)+\alpha^2(\cos(2\theta)+\cos(2\omega)))} \qquad (2.7)$$

$$R_y(\ell) = \frac{\alpha^2}{2\sin^2\theta}\left(\frac{\cos(\ell\theta)}{1-\alpha^2} - \frac{\cos((2+\ell)\theta)-\alpha^2\cos(\ell\theta)}{1+\alpha^4-2\alpha^2\cos(2\theta)}\right) \quad (\ell \geq 0) \qquad (2.8)$$

The plots of these functions for the various pole magnitudes and angles used are given in Appendix E. The $\theta = 0$ case is equivalent to having 2 poles on the real axis at a given magnitude in the Z plane. As would be expected, the power spectrum for each model showed higher magnitudes at digital frequencies close to the pole angle. Higher pole magnitudes yielded sharper, more well-defined power spectrum magnitudes at the given frequency, and lower pole magnitudes yielded less well defined more spread-out power spectra. Low pole magnitudes almost completely obliterated evidence of low frequency power spectrum components, and degraded its definition and sharpness at higher frequencies. This corresponds to the observed results in the image textures generated. The autocorrelation functions also reflect the appropriate relationship to the power spectra as outlined in the discussion for the two pole case, i.e., greater variation in the autocorrelation function indicates greater variation between pixels a given distance apart, which in turn implies higher spatial frequencies.

2. Two Real Poles

Rather than using complex conjugate pole locations to obtain real filter coefficient values, two poles on the real axis may also be used for a given direction. They may be placed at different locations on the real axis, or they may be placed together. The latter situation is equivalent to the $\theta = 0$ (or $\theta = \pm\pi$ if placed on the negative real axis) case, as mentioned above. For the two real pole case, the relevant equations are:

$$H(z) = \frac{1}{(1-\alpha_a z^{-1})(1-\alpha_b z^{-1})} = \frac{1}{1-(\alpha_a+\alpha_b)z^{-1}+\alpha_a\alpha_b z^{-2}}$$

$$= \frac{1}{1+a_{01}z^{-1}+a_{02}z^{-2}}$$

where

$$a_{01} = -(\alpha_a + \alpha_b)$$

$$a_{02} = \alpha_a \cdot \alpha_b$$

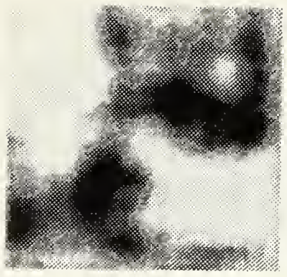
For these experiments a transfer function of the complex conjugate pole form was used for the z_1 direction, with $\alpha_1 = 0.9$ and $\theta_1 = 0$. For the z_2 direction a transfer function with two poles on the real axis was used. The image textures that result for various values of α_a and α_b are given in Figures 2-14 and 2-15. Some observations can be made about these results:

1) With poles placed at the same value on the z_2 real axis, rather unstructured, low frequency variations are observed in the z_2 direction. The more positive poles result in very slow variation in the image texture, while the lower magnitude positive poles show more variation in the z_2 direction. When the poles are placed in the same location on the negative side of the real axis, a low contrast image with some noticeable high frequency variations results.

2) As the poles are moved farther apart on the z_2 real axis, high frequency variations with increasing structure and oscillatory form are evidenced in the z_2 direction.

3) When poles with equal magnitude and opposite sign are used, fairly structured high frequency variations are evidenced in certain areas of the image, while low frequency variations are evident in other areas in the z_2 direction. Higher magnitude poles yield more discernible, structured variations, while lower magnitude poles of opposite sign yield discernible but non-oscillatory high frequency variations in certain areas of the image.

Of particular interest is the fact that two poles placed at the same value on the negative real axis in the z_2 plane yielded some high frequency variations. This is in keeping with the fact that values on the negative real axis correspond to a pole angle (and corresponding digital frequency) of $\theta = \pi$. The presence of poles on the negative side of the real axis of the z_2 plane seems to give rise to the high frequency variations with gradually more structure and oscillatory appearance as the pole is moved to the left (more negative).

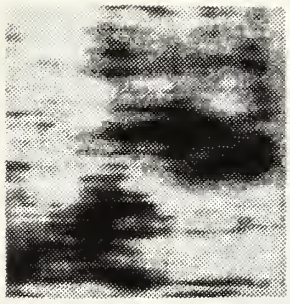


$\alpha_a = 0.9 \quad \alpha_b = 0.9$

$\alpha_a = 0.5 \quad \alpha_b = 0.5$

$\alpha_a = -0.9 \quad \alpha_b = -0.9$

$\alpha_a = 0.9 \quad \alpha_b = 0.5$



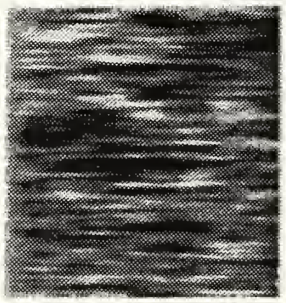
$\alpha_a = 0.9 \quad \alpha_b = 0.0$

$\alpha_a = 0.9 \quad \alpha_b = -0.2$

Figure 2-14 Images Generated Using a Four Pole (Two Real Poles) Autoregressive Model

$\alpha_a = 0.8 \quad \alpha_b = -0.9$

$\alpha_a = 0.9 \quad \alpha_b = -0.9$



$\alpha_a = 0.5 \quad \alpha_b = -0.5$

Figure 2-15 Images Generated Using a Four Pole (Two Real Poles) Autoregressive Model

The expressions for the power spectrum and autocorrelation function for the random process produced by driving the filter of Figure 2-10 with white noise are derived in Appendix F. It is shown there that the power spectral density and the autocorrelation function are given by:

$$S_y(\omega) = \frac{1}{1-2(\alpha_a+\alpha_b+\alpha_a^2\alpha_b+\alpha_a\alpha_b^2)\cos(\omega)+2\alpha_a\alpha_b\cos(2\omega)+\alpha_a^2 + 2\alpha_a\alpha_b+\alpha_b^2+\alpha_a^2\alpha_b^2} \qquad (2.9)$$

$$R_y(\ell) = \frac{1}{(\alpha_a-\alpha_b)^2}\left[\frac{\alpha_a^{2+\ell}}{1-\alpha_a^2} - \frac{\alpha_a^{\ell+1}\alpha_b+\alpha_a\alpha_b^{\ell+1}}{1-\alpha_a\alpha_b} + \frac{\alpha_b^{2+\ell}}{1-\alpha_b^2}\right] \qquad (2.10)$$

Plots of these functions for the various values used in this section are given in Appendix G. The power spectrum results are consistent with the observed image spatial frequency characteristics. Both low and high frequency components were contained in some of the power spectra, and were manifested in the corresponding images as both low and high frequency variations in the z_2 direction. The nature of the autocorrelation functions related to the power spectra that contained low and high frequency components was interesting. Autocorrelation functions with much variation but all positive values, rather than the equal magnitude positive and negative values evidenced in earlier results, seems to reflect the higher level of correlation related to the low frequency (smoother variations) aspect of the image texture variations.

D. IMAGE TEXTURE ROTATION TRANSFORMATION

If an image signal $x(n_1,n_2)$ consists of a rotated version of another image $w(m_1,m_2)$ such that $n_1 = Im_1 + Jm_2$ and $n_2 = Km_1 + Lm_2$, where I, J, K, and L are integers and $IL-KJ \neq 0$, then the Z transform $X(z_1,z_2)$ is given by $W(z_1^I,z_2^K,z_1^J,z_2^L)$ [Ref. 3:p. 182]. A 45^o rotation corresponds to $I = 1$, $K = 1$, $J = 1$, $L = -1$. If we use the four pole separable result for $H(z_1,z_2)$, as shown in Figure 2-10, and apply the above transformation $(z_1 \rightarrow z_1^1 z_2^1, z_2 \rightarrow z_1^1 z_2^{-1})$, we find after simplification:

$$H_R(z_1 z_2) = \frac{1}{\begin{array}{c}1+a_{01}z_1^{-1}z_2^{-1}+a_{02}z_1^{-2}z_2^{2}+a_{10}z_1^{-1}z_2^{-1}+a_{10}a_{01}z_1^{-2}+a_{10}a_{02}z_1^{-3}z_2^{-1} \\ +\, a_{20}z_1^{-2}z_2^{-2}+a_{20}a_{01}z_1^{-3}z_2^{-1}+a_{20}a_{02}z_1^{-4}\end{array}} \qquad (2.11)$$

Notice that this transfer function is not separable but consists of a rotated version of a separable filter. Figure 2-16 illustrates the support of the denominator polynomial for this filter. It has the form of a non-symmetric half-plane infinite impulse response (IIR) filter, so it is recursively computable.

0	0	a_{20}	0	
0	$a_{20}a_{01}$	0	a_{10}	
$a_{20}a_{02}$	0	$a_{10}a_{01}$	0	1
0	$a_{10}a_{02}$	0	a_{01}	0
0	0	a_{02}	0	0

Figure 2-16 Rotation Transformation Filter Form

The application of this filter, using filter coefficients of the four pole separable filter with poles at $z_1 \rightarrow 0.9e^{\pm j\frac{\pi}{2}}$, $z_2 \rightarrow 0.9e^{\pm j0}$ in the original separable filter yielded the result shown in Figure 2-17.

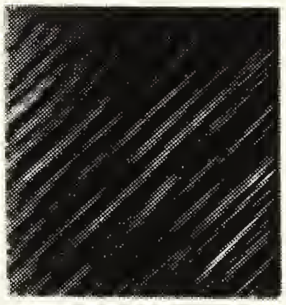


Figure 2-17 Result of Rotation Transformation

E. SUMMARY

Autoregressive models can produce a variety of image textures. For general two-dimensional models, the system functions are generally not factorable and singularities occur on surfaces, not at isolated points. For these reasons it is difficult to design two-dimensional filters for images and predict the resulting character of the images. Indeed, even to ensure stability of the filter is not trivial. As a result we concentrated here on separable forms, which by their nature are much easier to analyze. Certain types of texture patterns using various separable autoregressive models can be predicted based on filter pole placement in the z_1 and z_2 planes. Arbitrary or random selection of filter coefficients can yield interesting but generally unpredictable results. Obviously, an infinite number of variations on the models above could be attempted. Ultimately, the anticipated utility of the textures generated will guide the process of model and parameter selection.

III. IMPLEMENTATION OF AN FIR SUMMATION FILTER IN TWO DIMENSIONS

To implement the 2-D ARIMA model, the inverse of a filter representing a suitable difference operator is needed. One possible 2-D difference operator is the Laplacian, which has the impulse response shown in Figure 3-1 [Ref. 8:pp. 212-213].

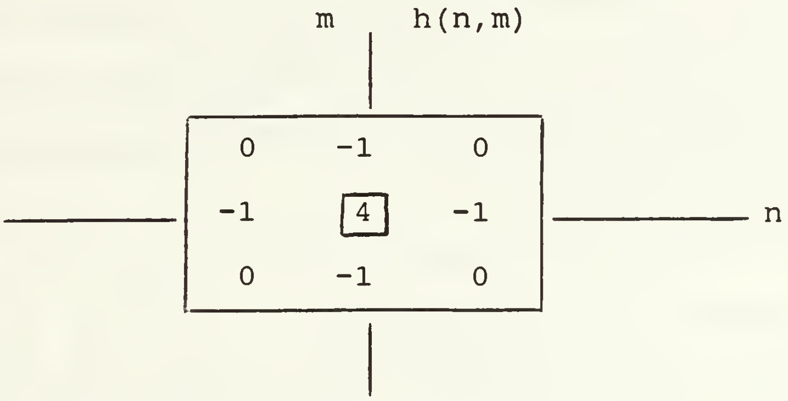


Figure 3-1 Laplacian Impulse Response

Its implementation involves convolving it with an image and is represented by the following difference equation:

$$y(n,m) = \sum_{i=-1}^{1} \sum_{j=-1}^{1} b_{ij} x(n-i,m-j) \qquad (3.1)$$

where

$x(n,m)$ is the image input signal, and

b_{ij} is the filter coefficient matrix ($b_{ij} = h(i,j)$)

In the Z-transform domain this can be written as:

$$\begin{aligned} Y(z_1,z_2) &= H(z_1,z_2)\cdot X(z_1,z_2) \\ &= (4 - z^{-1} - z_1 - z^{-1} - z_2)\cdot X(z_1,z_2) \end{aligned}$$

In areas of an image where adjacent pixels have similar gray levels (low frequency, homogeneous areas), the result of this operator will be approximately zero. Where significant or sharp differences in gray levels between adjacent pixels exist, the result of this operation will be farther from zero. Thus the Laplacian difference operator is sometimes used as an "edge detector."

The problem addressed in this chapter is constructing the inverse of the operator. In the Z domain, the expression for the inverse would be [Ref. 4:p. 36]:

$$H^{-1}(z_1,z_2) = \frac{1}{H(z_1,z_2)} = \frac{1}{4 - z_1^{-1} - z_1 - z_2^{-1} - z_2}$$

which has an expansion as an infinite series of positive and negative powers of z_1 and z_2. That is, considering this expression as a problem in long division, the result of such division would be an expression of the form:

$$H^{-1}(z_1,z_2) = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} a_{ij} z_1^{-i} z_2^{-j}$$

where

a_{ij} = coefficient values of $z_1^{-i}z_2^{-j}$ resulting from the long division

Note that if

$$\frac{1}{4 - z_1^{-1} - z_1 - z_2^{-1} - z_2} = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} a_{ij} z_1^{-i} z_2^{-j}$$

then cross multiplication would yield:

$$(4 - z^{-1} - z_1 - z^{-2} - z_2) \cdot \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} a_{ij} z^{-i} z^{-j} = 1 \qquad (3.3)$$

The double summation expression in z_1 and z_2 will be truncated and considered to be an FIR filter with finite support and coefficients a_{ij}. This approximates the desired inverse filter. In particular, we will use the following constraints:

1) Choose the limits of summation to be equal in both directions, i.e.,

$$\sum_{i=-L}^{L} \sum_{j=-L}^{L} a_{ij} z_1^{-i} z_2^{-j}$$

This results in a "square" region of support for the filter (all values outside assumed zero).

2) Force the values for the filter coefficients to be symmetric, i.e., $a_{ij} = a_{-ij} = a_{i-j} = a_{-i-j} = a_{ji} = a_{-ji} = a_{j-i} = a_{-j-i}$.

Using these constraints and implementing the cross multiplication equation (3.3) will result in an expression in z_1 and z_2, with each combination of the $z_1^{-i}z_2^{-j}$ terms having a coefficient whose form is a summation of terms in a_{ij}

where the coefficients of a_{ij} are either 4 or -1. The coefficient of the $z_1^0 z_2^0$ term must equal 1 and the coefficient of any other $z_1^{-i} z_2^{-j}$ term must equal zero to satisfy equation (3.3).

As an example, let L = 1. Equation (3.3) can then be expressed as:

$$(4 - z^{-1} - z_1 - z^{-1} - z_2) \cdot \sum_{i=-1}^{1} \sum_{j=-1}^{1} a_{ij} z_1^{-i} z_2^{-j}$$
$$= 1 + 0 \cdot z_1^1 + 0 \cdot z_2^1 + 0 \cdot z_1^1 z_2^1 + \ldots$$

Performing the double summation yields

$$\sum_{i=-1}^{1} \sum_{j=-1}^{1} a_{ij} z_1^{-i} z_2^{-j} = a_{-1-1} z_1^1 z_2^1 + a_{-10} z_1^1 z_2^0 + a_{-11} z_1^1 z_2^{-1} +$$
$$+ a_{00} z_1^0 z_2^0 + a_{0-1} z_1^0 z_2^1 + a_{1-1} z_1^{-1} z_2^1 + a_{10} z_1^{-1}$$
$$+ a_{01} z_2^{-1} + a_{11} z_1^{-1} z_2^{-1}$$

Performing the cross multiplication would yield 45 different terms in various combinations of $z_1^{-i} z_2^{-j}$. Combining these terms to find the coefficient expression for each $z_1^{-i} z_2^{-j}$ term soon becomes rather tedious and impractical for even moderate values of L. An alternative way to proceed is to choose a $z_1^a z_2^b$ term on the right hand side of the equation, and for each term in the expression $4 - z_1^{-1} - z_1 - z_2^{-1} - z_2$, determine what values of i and j are required so that when each term is multiplied by $a_{ij} z_1^{-i} z_2^{-j}$, it will result in an expression in the chosen $z_1^a z_2^b$ term on the right hand side of the equation. Choosing $z_1^0 z_2^0$ (= 1) on the right hand side of the equation, we have:

$$4 \cdot a_{ij} z_1^{-i} z_2^{-j} = c z_1^0 z_2^0 \text{ when } i = 0 \text{ and } j = 0; \text{ so } c = 4a_{00}$$
$$-z_1^{-1} \cdot a_{ij} z_1^{-i} z_2^{-j} = c z_1^0 z_2^0 \text{ when } i = -1 \text{ and } j = 0; \text{ so } c = -a_{-10}$$
$$-z_1 \cdot a_{ij} z_1^{-i} z_2^{-j} = c z_1^0 z_2^0 \text{ when } i = 1 \text{ and } j = 0; \text{ so } c = -a_{10}$$
$$-z_2^{-1} \cdot a_{ij} z_1^{-i} z_2^{-j} = c z_1^0 z_2^0 \text{ when } i = 0 \text{ and } j = -1; \text{ so } c = -a_{0-1}$$
$$-z_2 \cdot a_{ij} z_1^{-i} z_2^{-j} = c z_1^0 z_2^0 \text{ when } i = 0 \text{ and } j = 1; \text{ so } c = -a_{01}$$

So the coefficient for $z_1^0 z_2^0$ is simply a summation of the c terms obtained above, i.e.,

$$(4a_{00} - a_{-10} - a_{10} - a_{0-1} - a_{01}) z_1^0 z_2^0$$

This entire expression must equal 1 to satisfy (3.3), and since $z_1^0 z_2^0 = 1$, $4a_{00} - a_{-10} - a_{10} - a_{0-1} - a_{01} = 1$ also.

Using the same method for the $z_1^1 z_2^1$ term on the right hand side yields:

$$4 \cdot a_{ij} z_1^{-i} z_2^{-j} = c z_1^1 z_2^1 \text{ when } i = -1 \text{ and } j = -1; \text{ so } c = 4a_{-1-1}$$
$$-z_1^{-1} \cdot a_{ij} z_1^{-i} z_2^{-j} = c z_1^1 z_2^1 \text{ when } i = -2 \text{ and } j = -1; \text{ so } c = -a_{-2-1}$$
$$-z_1 \cdot a_{ij} z_1^{-i} z_2^{-j} = c z_1^1 z_2^1 \text{ when } i = 0 \text{ and } j = -1; \text{ so } c = -a_{0-1}$$
$$-z_2^{-1} \cdot a_{ij} z_1^{-i} z_2^{-j} = c z_1^1 z_2^1 \text{ when } i = -1 \text{ and } j = -2; \text{ so } c = -a_{-1-2}$$
$$-z_2 \cdot a_{ij} z_1^{-i} z_2^{-j} = c z_1^1 z_2^1 \text{ when } i = -1 \text{ and } j = 0; \text{ so } c = -a_{-10}$$

The resulting term in $z_1^1 z_2^1$ is:

$$(4a_{-1-1} - a_{-2-1} - a_{0-1} - a_{-1-2} - a_{-10}) z_1^1 z_2^1$$

This expression must equal zero, since there is no $z_1^1 z_2^1$ term on the right side of (3.3), so $4a_{-1-1} - a_{-2-1} - a_{0-1} - a_{-1-2} - a_{-10} = 0$.

This procedure can be extended to any number of $z_1^{-i} z_2^{-j}$ terms. When this is done, the resulting expressions for the coefficients of $z_1^{-i} z_2^{-j}$ can be formed into a set of simultaneous equations in order to solve for the a_{ij} coefficient values. However, due to the symmetry condition imposed above, some of the equations for the coefficients of the $z_1^{-i} z_2^{-j}$ terms are linearly dependent. For values of i and j that yield unique or distinct values for a_{ij}, the resulting $z_1^{-i} z_2^{-j}$ coefficient expressions are linearly independent. For example, the coefficient expression for the $z_1^{-1} z_2^{0}$ term is linearly independent of the coefficient expression for the $z_1^{-1} z_2^{-0}$ term, since $a_{10} \neq a_{11}$. But the coefficient expression for the $z_1^{1} z_2^{0}$ term is linearly dependent on the coefficient expression for the $z_1^{0} z_2^{-1}$ term, since $a_{10} = a_{01}$. Using only the linearly independent equations for a given filter size yields a set of p equations in p unknowns, where p is the number of unique and distinct a_{ij} values in the inverse filter. The value of p is related to the size of the desired inverse filter. If the size of the filter is $N \times N$, the number of unique a_{ij} values using the symmetry of constraint above is:

$$p = (\frac{N-1}{2}+1) + (\frac{N-1}{2}) + (\frac{N-1}{2}-1) + \ldots + 1 \qquad (3.4a)$$

$$= \frac{(N+1)(N+3)}{4} \qquad (3.4b)$$

For example, with N = 7, the unique a_{ij} values in a 7×7 inverse filter can be represented by a_{33}, a_{32}, a_{31}, a_{30}, a_{22}, a_{21}, a_{20}, a_{11}, a_{10}, a_{00}. Though there are 49 elements in a 7×7 filter, all of them are equal to one of these values listed, due to symmetry. Obviously, N is constrained to be odd for a square filter with a unique element a_{00} in the center.

The solution of the resulting p equations yields the values for the p filter elements or coefficients. This defines the FIR approximation to the inverse filter. It is only an approximation due to the finite size constraint imposed, and it might be expected that the larger the filter size, the better the approximation.

An algorithmic procedure for obtaining the a_{ij} coefficients is outlined below. An example follows.

1) Determine the desired size of the inverse filter.

2) For each combination of (positive) i,j values corresponding to a <u>unique</u> a_{ij} filter coefficient, identify the five term summation equation associated with each $z_1^{-i}z_2^{-j}$ term.

3) Combine equal a_{ij} values and develop a matrix of coefficients for the a_{ij} values. Let this matrix be $\underline{A}$.

4) Denoting the vector of unique a_{ij} values as $\bar{a}$, the set of simultaneous equations in matrix form is:

$$\bar{A}\bar{a}^T = \begin{bmatrix} 1 \\ 0 \\ \cdot \\ \cdot \\ \cdot \\ 0 \end{bmatrix} \tag{3.5}$$

where

$$\bar{a} = [a_{00} \quad a_{10} \quad a_{11} \quad a_{20} \quad a_{21} \quad a_{22} \cdots a_{\frac{N-1}{2}\frac{N-1}{2}}]$$

The top row of $\bar{A}$ corresponds to the summation of a_{ij} terms that represents the coefficient of the $z_1^0 z_2^0$ term.

5) Solve (3.5) for $\bar{a}$.

An example of this procedure is appropriate at this point. For an inverse filter of size $N \times N$:

Step 1

Let $N = 5$ (therefore $L = 2$)

Step 2

The coefficients corresponding to each unique $z^{-i}z^{-j}$ term are:

$$z_1^0 z_2^0 \rightarrow 4a_{00} - a_{10} - a_{-10} - a_{01} - a_{0-1}$$
$$z_1^{-1} z_2^0 \rightarrow 4a_{10} - a_{20} - a_{00} - a_{11} - a_{1-1}$$
$$z_1^{-1} z_2^{-1} \rightarrow 4a_{11} - a_{21} - a_{01} - a_{12} - a_{10}$$

$$z_1^{-2}z_2^{0} \rightarrow 4a_{20} - a_{30} - a_{10} - a_{21} - a_{2-1}$$
$$z_1^{-2}z_2^{-1} \rightarrow 4a_{21} - a_{31} - a_{11} - a_{20} - a_{2-2}$$
$$z_1^{-2}z_2^{-2} \rightarrow 4a_{22} - a_{32} - a_{12} - a_{23} - a_{21}$$

Step 3

Combining equal terms in Step 2 and expressing the coefficients of a_{ij} in matrix form yields an $\bar{A}$ matrix of:

$$\bar{A} = \begin{bmatrix} 4 & -4 & 0 & 0 & 0 & 0 \\ -1 & 4 & -2 & -1 & 0 & 0 \\ 0 & -2 & 4 & 0 & -2 & 0 \\ 0 & -1 & 0 & 4 & -2 & 0 \\ 0 & 0 & -1 & -1 & 4 & -1 \\ 0 & 0 & 0 & 0 & -2 & 4 \end{bmatrix}$$

Step 4

With $\bar{A}$ given in Step 3, the $\bar{a}$ vector for (3.5) is

$$\bar{a} = [a_{00} \quad a_{10} \quad a_{11} \quad a_{20} \quad a_{21} \quad a_{22}]$$

Step 5

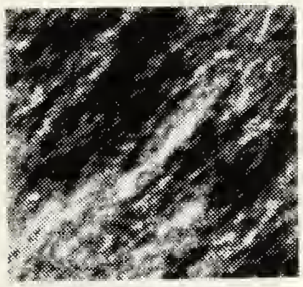
Solving (3.5) for $\bar{a}$ involves inverting $\bar{A}$ and multiplying it by $[1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0]^T$. Thus:

$$\bar{a} = \bar{A}^{-1} \cdot [1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0]^T$$

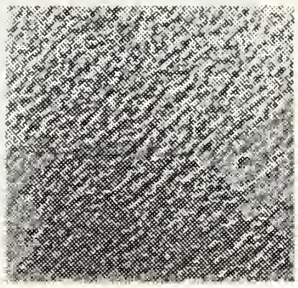
Appendix H illustrates the forms of the resulting inverse filters of various sizes, as well as the normalized and unnormalized filter cross sections.

One way to validate the resulting inverse filter is to convolve it with the original Laplacian difference operator. The result should approximate an impulse at the origin. Appendix I shows the results of this convolution using 3×3, 5×5, 7×7, 9×9, 15×15, and 21×21 size inverse filters. It is seen there that as the size of the filter gets larger, it becomes a better approximation to the true inverse and the convolution looks more like an impulse.

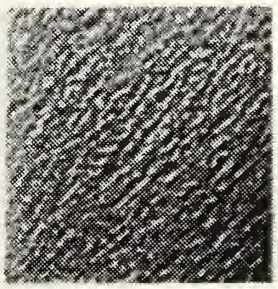
To test the application of this filter on an actual image, a test image was filtered using the Laplacian difference operator. Then the resulting image data were filtered again using various size inverse filters. The results are shown in Figures 3-2 and 3-3. Note that the image resulting from Laplacian FIR filtering seems more stationary than the test image, which was one of the desired results. Inverse filtering of that result yields images that are progressively more similar to the original test image as the size of the inverse filter increases. However, a rather large inverse filter is needed to accurately reproduce the image. The result of the 21×21 inverse filter is quite similar to the test image, with some loss of contrast or darkness in certain areas, but with essentially the same pattern. The effect of the size limitation of the



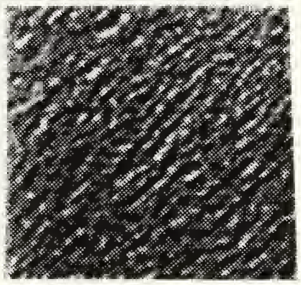
Test image



Laplacian filtered test image

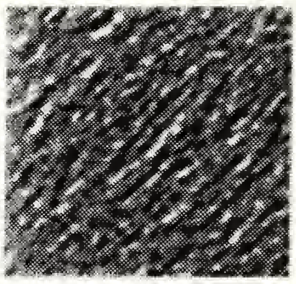


3 × 3 inverse filter

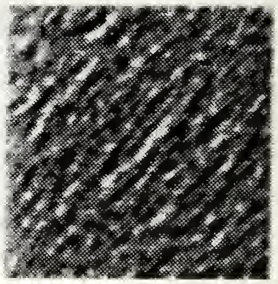


5 × 5 inverse filter

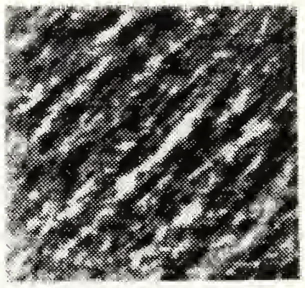
Figure 3-2 Results of Filtering Test Image with Laplacian FIR Filter and its Inverse



7×7 inverse filter



9×9 inverse filter



15×15 inverse filter

21×21 inverse filter

Figure 3-3 Results of Filtering Test Image with Laplacian FIR Filter and its Inverse

inverse filter, as manifested in the convolution of the Laplacian and its inverse above, would seem to explain the lack of perfect test image reproduction. Larger inverse filter sizes could be tried, but large inverse filter sizes relative to image size would result in a significant portion around the edge of the image having only a part of the filter applied to it. This would adversely affect the overall quality of image reproduction.

IV. APPLICATION OF THE ARIMA MODEL TO IMAGE TEXTURES

As outlined in Chapter I, the utility of the ARIMA model centers around the fact that a difference operator applied to an image texture may improve the stationarity of the image statistical characteristics. A stationary image texture is required for accurate modeling by autoregressive techniques, and it was hoped that application of the autoregressively generated texture to an approximate inverse of the difference operator may yield a more accurate or recognizable representation of the original nonstationary image, as compared to a purely autoregressively generated version.

A. APPLICATION OF LAPLACIAN INVERSE FILTER TO AUTO-REGRESSIVELY GENERATED IMAGES

As an initial examination of the effects of the inverse filter developed in Chapter III on image textures, selected images generated in Chapter II were input to the 21×21 version of that filter. Figures 4-1 through 4-3 illustrate the results. All attempts resulted in a blurred or smoothed version of the original image. Since the inverse of a difference operation is a summation or "integration" operation, and since integration operations can be expected to smooth or blur (low pass filter) signals [Ref. 8:pp. 136-154], the results are not surprising. However,

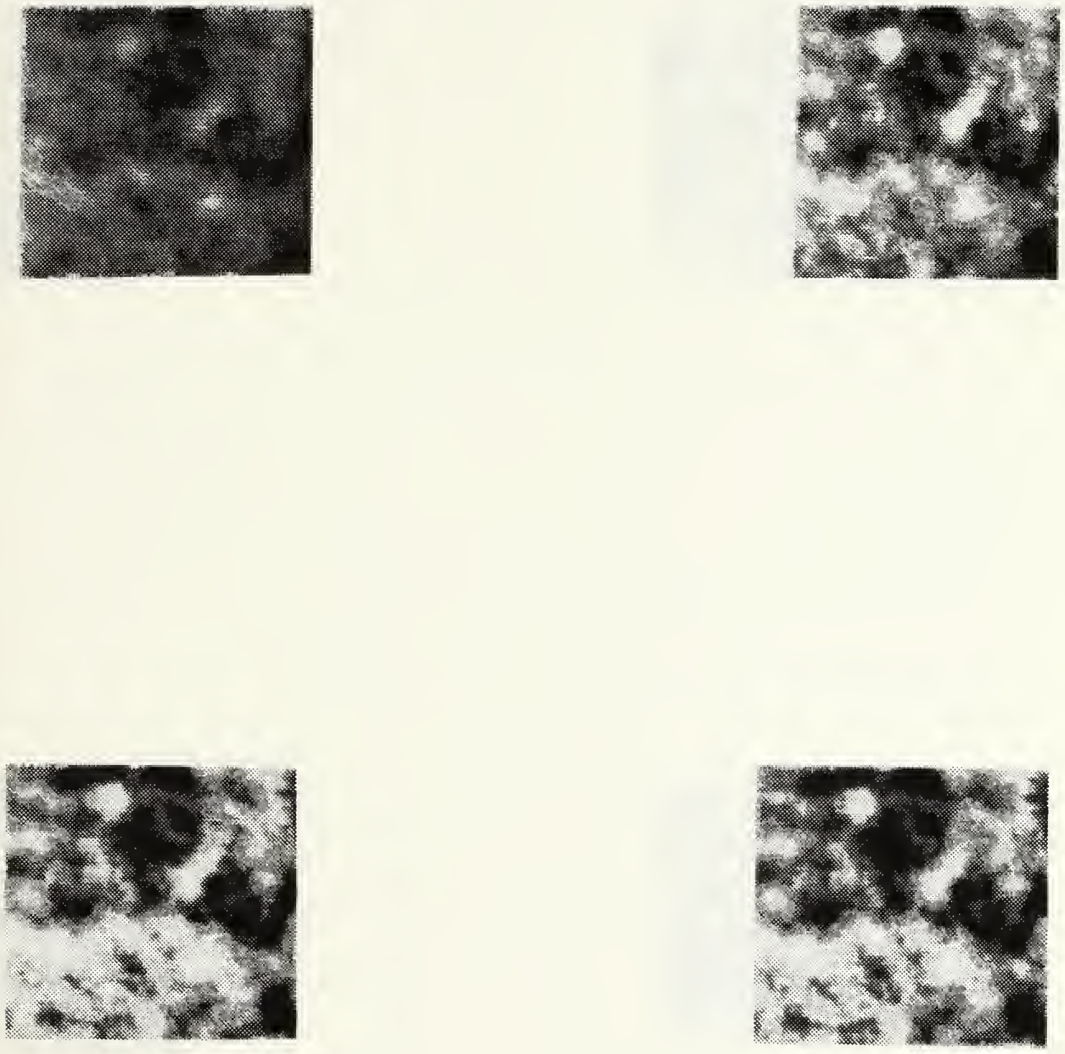


Figure 4-1 Images from Figure 2-7 (top 4) Applied to Summation Filter

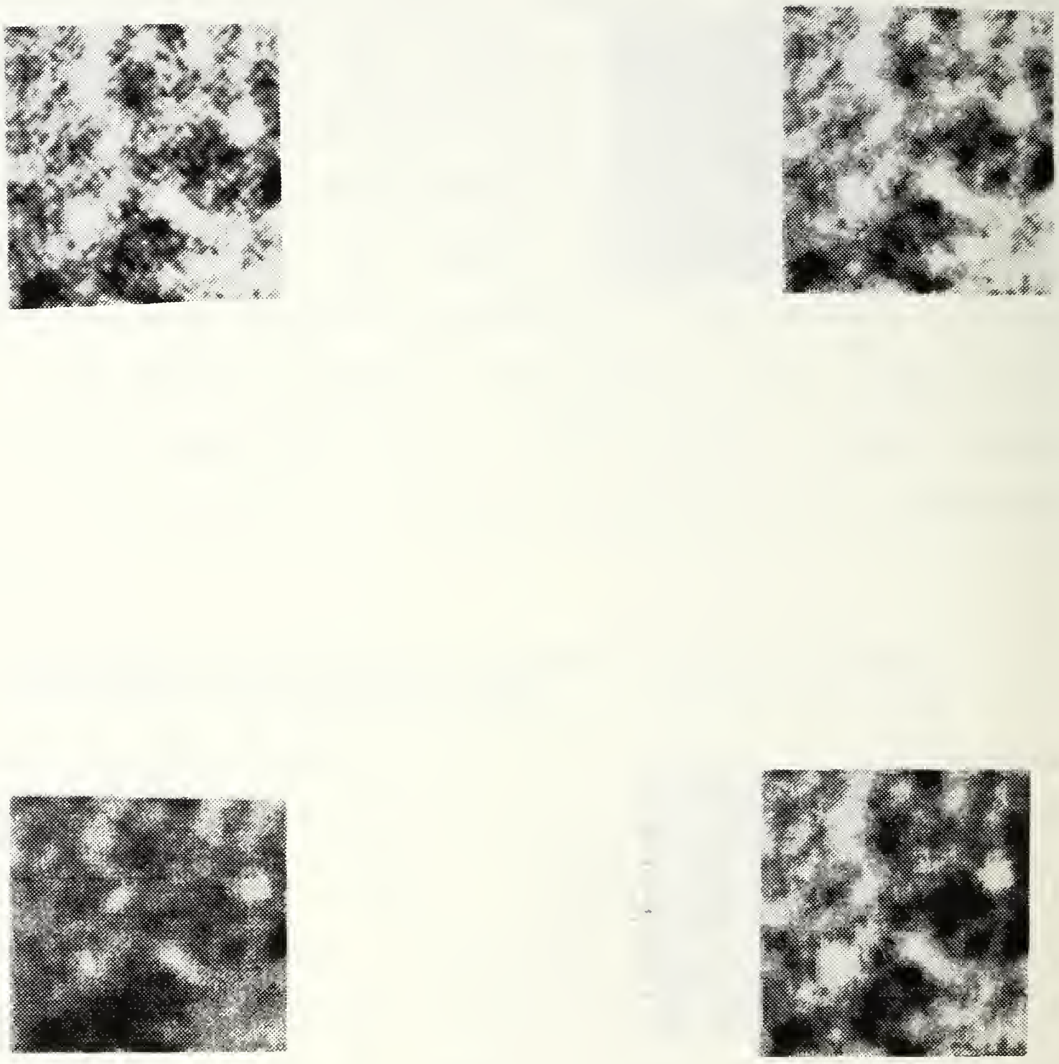


Figure 4-2 Images from Figure 2-11 (top 4) Applied to Summation Filter

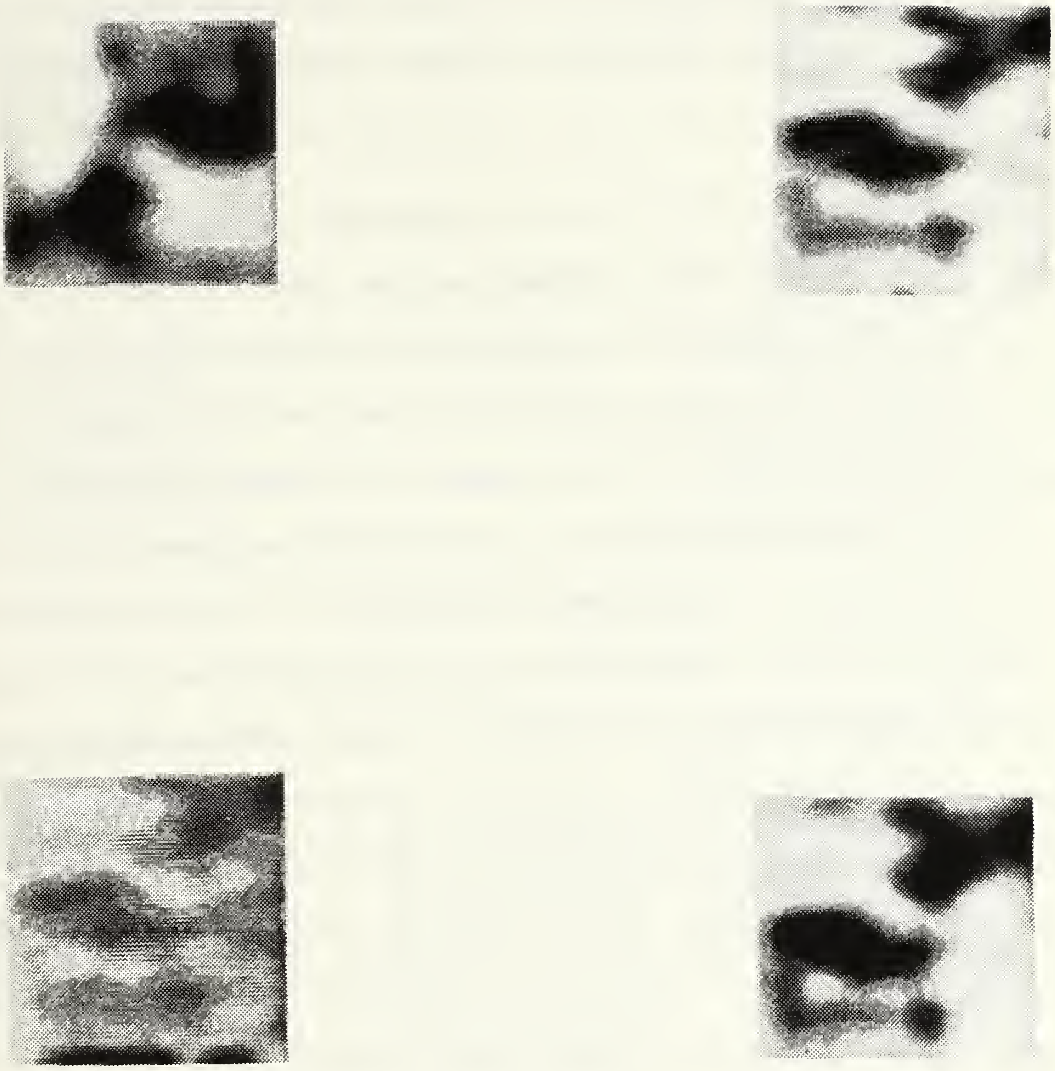


Figure 4-3 Images from Figure 2-14 (top 4) Applied to Summation Filter

except to the extent that blurring is useful or desirable, applying the summation filter to image signals that are not based on the application of the corresponding difference filter to that signal seems to be of little utility.

In the remainder of this chapter we consider application of the summation filter to regenerate actual image textures.

B. AUTOREGRESSIVE FILTER PARAMETER ESTIMATION PROCEDURES

The first step in testing the ARIMA model is to estimate the autoregressive, quarter plane filter parameters required to model the real image textures and the signal resulting from application of the Laplacian operator to those images. For a zero-mean signal, these model parameters are found by solving a set of Normal equations. In these equations the white noise covariance is referred to as the prediction error covariance. The Normal equations can be expressed as

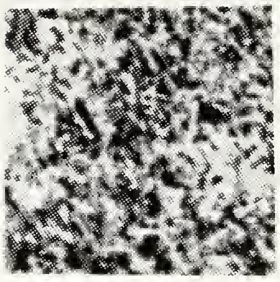
$$\bar{R} \begin{bmatrix} \bar{a}_0 \\ \bar{a}_1 \\ \cdot \\ \cdot \\ \cdot \\ \bar{a}_{p-1} \end{bmatrix} = \begin{bmatrix} \bar{s} \\ 0 \\ \cdot \\ \cdot \\ \cdot \\ 0 \end{bmatrix} \qquad (4.1)$$

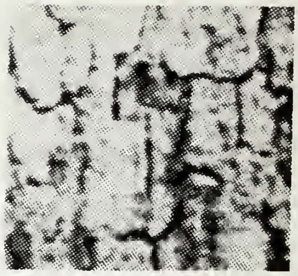
where the R matrix is the correlation matrix for the signal (block Toeplitz with Toeplitz blocks), the $\bar{a}$ vector consists of appropriately ordered filter coefficient vectors, and $\bar{s}$ is a vector containing the prediction error covariance as the first and only nonzero element. Here $\bar{a}_i = [a_{i0} \; a_{i1} \; a_{i2} \; . \; . \; . \; a_{i\,Q-1}]^T$ and $\bar{s} = [\sigma^2 \; 0 \; 0 \; . \; . \; . \; 0]^T$.

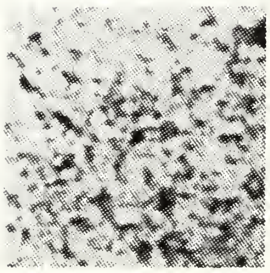
Calculating the correlation matrix and prediction error covariance from the image signals, and solving (4.1) for the $\bar{a}$ vectors, provides all the parameters needed for the 2-D AR model. The multichannel form of the Levinson recursion can be used to solve these equations more efficiently [Ref. 2:p. 454].

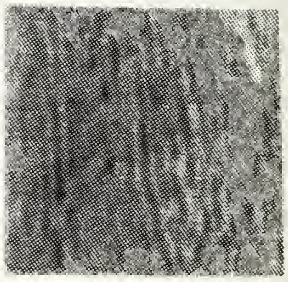
C. APPLICATION TO REAL IMAGE TEXTURES

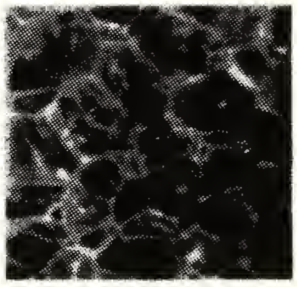
Actual image textures used here are from the image data base at the University of Southern California's Signal and Image Processing Institute [Ref. 9:pp. 13-14]. The images selected from this data base are contained in a book by Brodatz [Ref. 10]. Portions of the images of size 128×128 pixels were obtained and used as a basis for processing. Filter coefficients were calculated for the real image textures shown in Figures 4-4 and 4-5. Various filter sizes were tried to determine which yielded the best results in generating a particular image, and a quarter-plane filter size of 4×4 was selected. Results of autoregressive filtering of white noise using the appropriate calculated coefficients to model each texture are given in Figures 4-6 and 4-7. Images generated by applying the Laplacian difference operator to the real images are shown in Figures 4-8 and 4-9. Autoregressive generation of these images using


Grass


Bark


Sand


Water


Bubbles

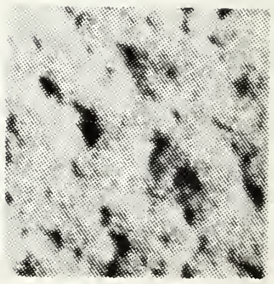
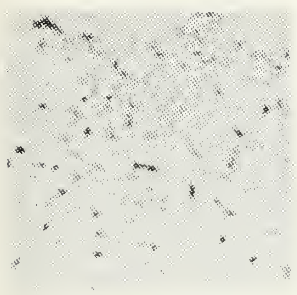
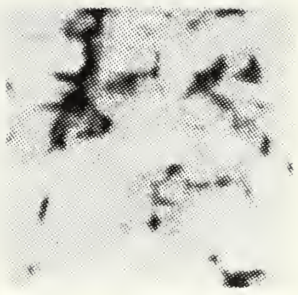

Wall

Figure 4-4 Actual Image Textures


Sand


Gravel

Grass

Figure 4-5 Actual Image Textures

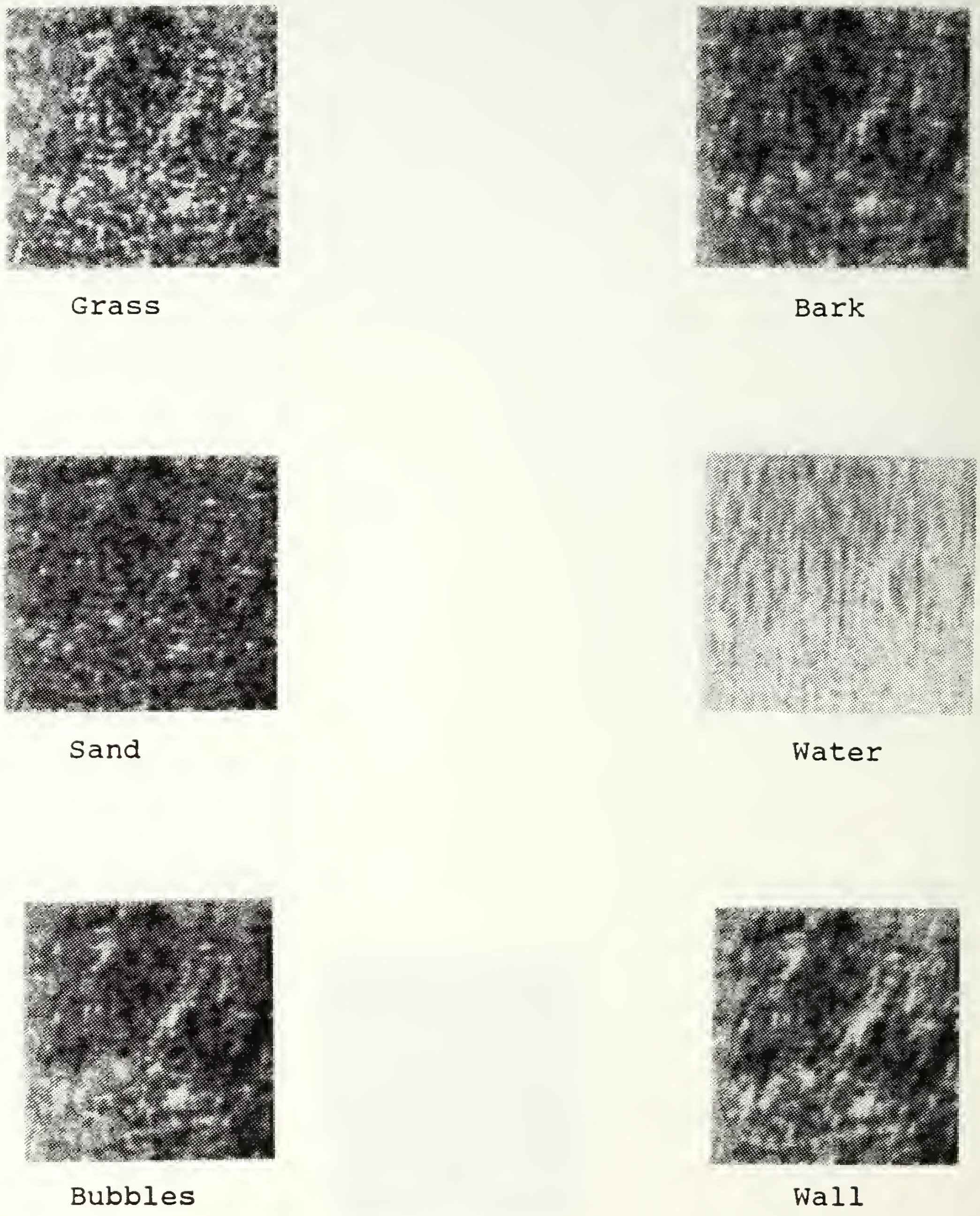


Figure 4-6 Image Textures Generated Using an AR Model

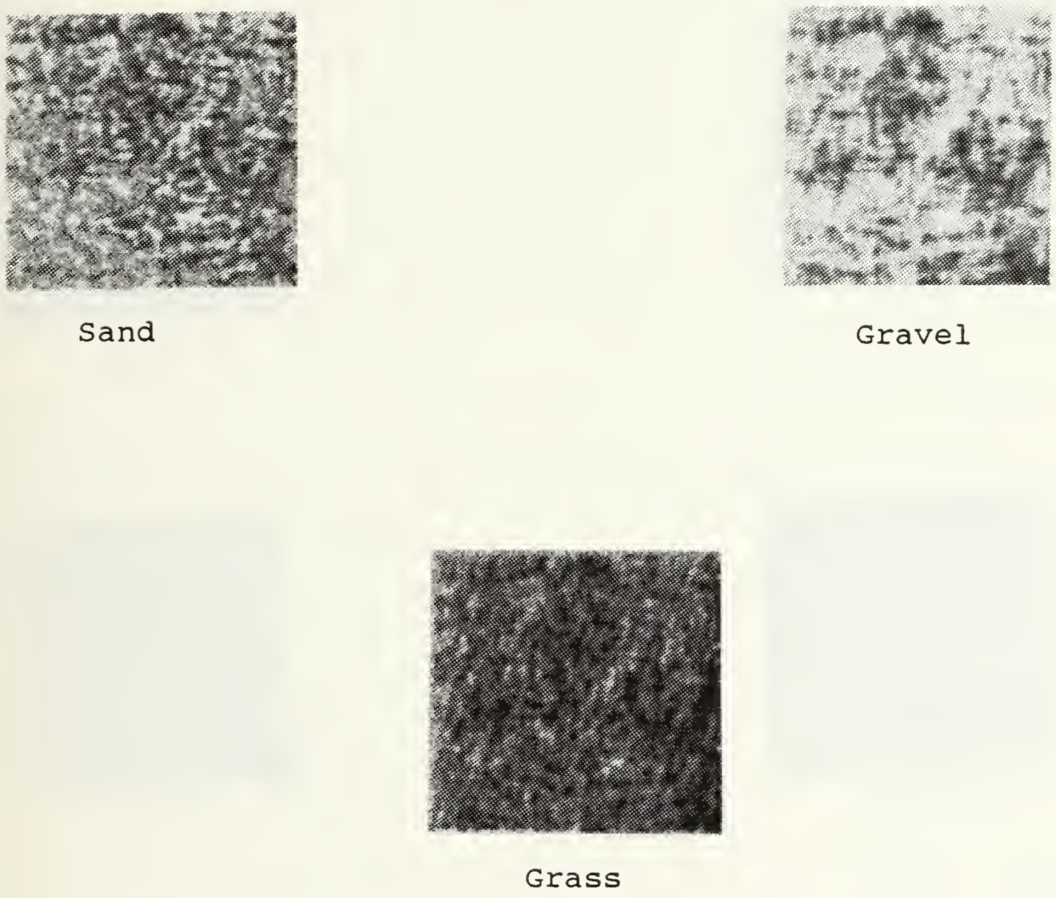


Figure 4 -7 Image Textures Generated Using an AR Model

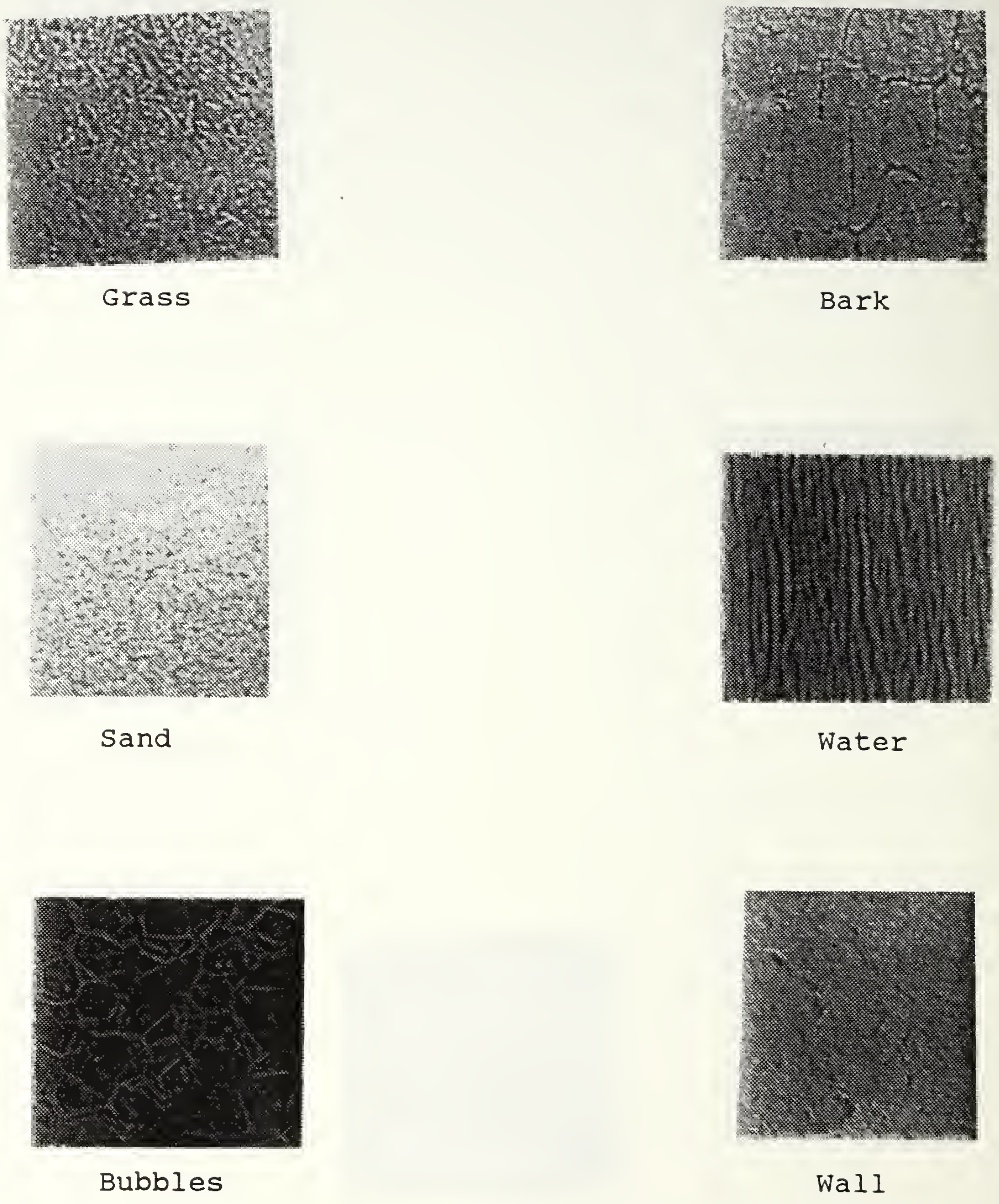
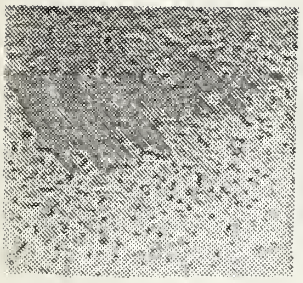
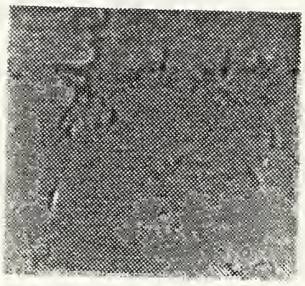


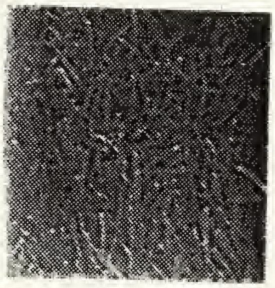
Figure 4-8 Actual Images After Laplacian Filtering



Sand



Gravel



Grass

Figure 4-9 Actual Images After Laplacian Filtering

filters with the corresponding calculated coefficients are given in Figures 4-10 and 4-11. Finally, the application of the signal represented by the images in Figures 4-10 and 4-11 (without the 0-255 scaling reflected in these figures) to the 21×21 inverse filter described in Chapter III yields the images shown in Figures 4-12 and 4-13. Comparison of all of the above results yields the following observations:

1) Autoregressive modeling of the water, grass and sand textures yielded good results. Some of the other textures with more structure and sharp local variations were not reproduced well.

2) Autoregressive reproduction of images created after application of the difference operator, with the exception of the water image, yielded generally poor results. As observed in Chapter III, the application of the difference operator produces a seemingly more stationary result, but the edge structure that remained in most of the images after application of the difference operator was in general not reproducible using a purely AR model.

3) Application of the inverse filter to the image signal generated by AR model reproduction of the difference operator results yielded smoothed versions of those results. This is similar to what was observed in Section A of this chapter when images were applied to the inverse filter that were not based on the specific data generated by the difference operator.

As a final test of the ARIMA model, a 64×64 contrast enhanced aerial photograph of trees, with smoother variations and in general less edge structure than the other images tested, was tried. The results are shown in Figure 4-14. Though this image seemed somewhat better adapted to the model, overall the same observations outlined above apply.

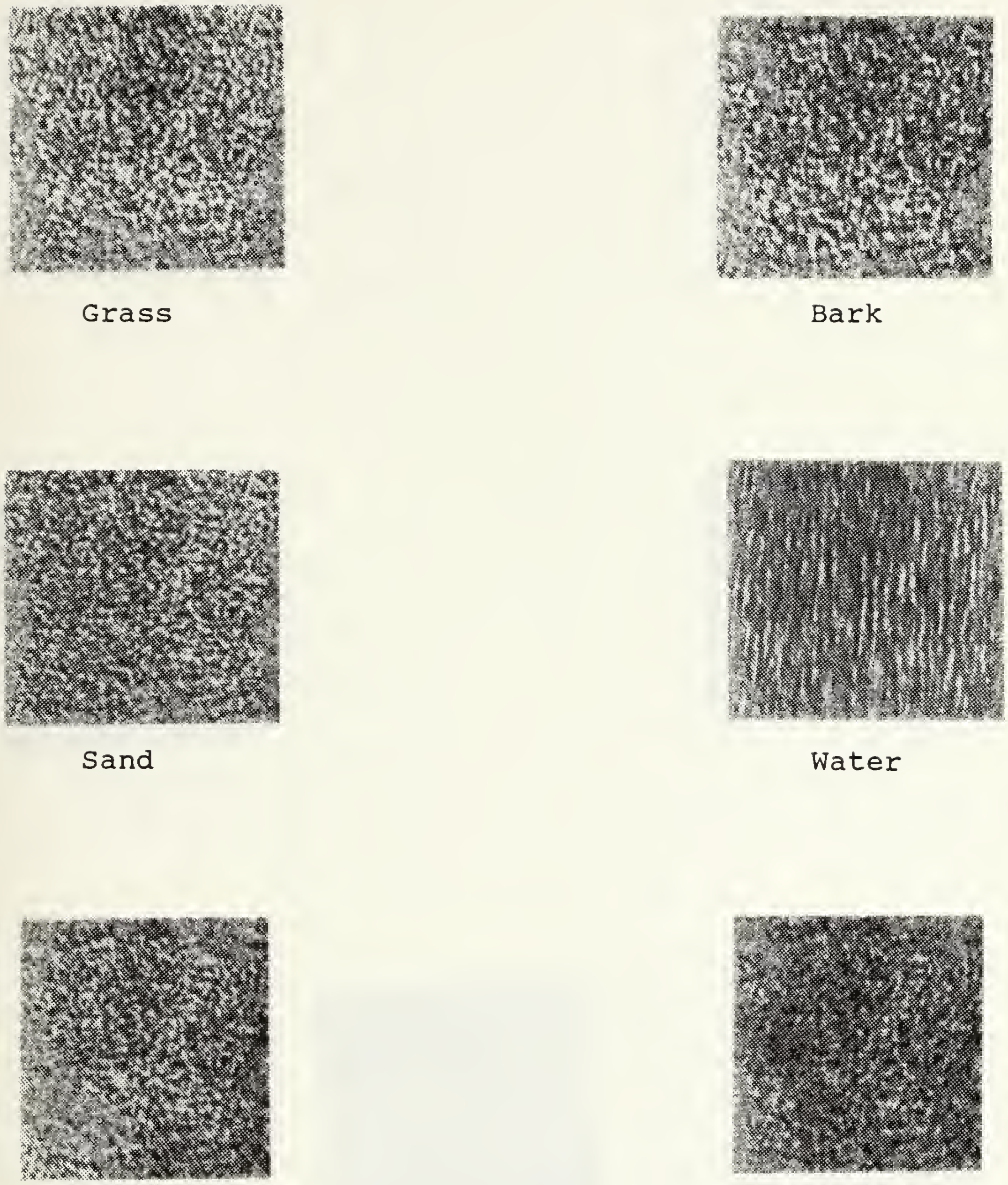
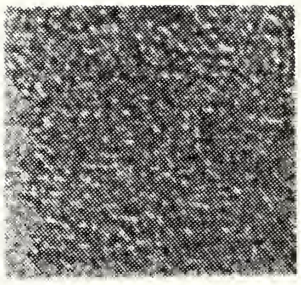
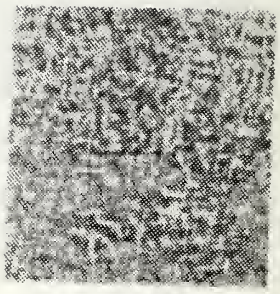


Figure 4-10 Laplacian Filtered Image Textures Generated Using an AR Model


Sand


Gravel

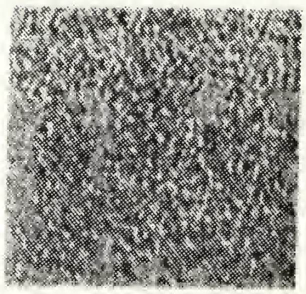

Grass

Figure 4-11 Laplacian Filtered Image Textures Generated Using an AR Model

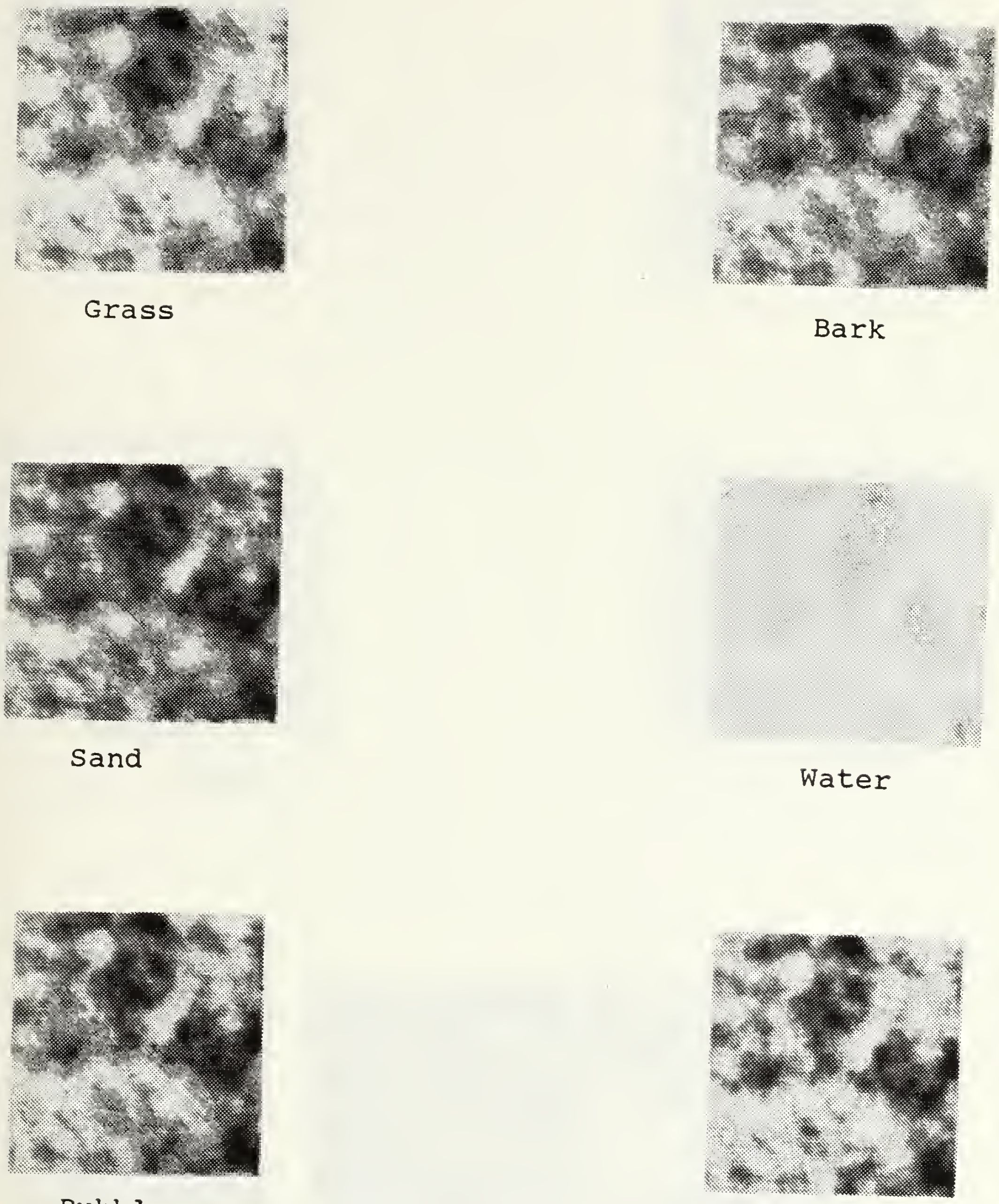
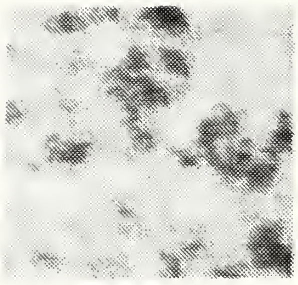
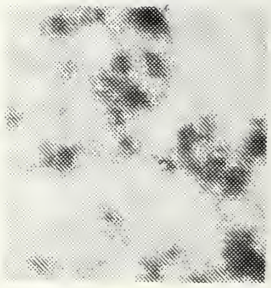


Figure 4-12 Image Textures Generated Using an ARIMA Model


Sand


Gravel

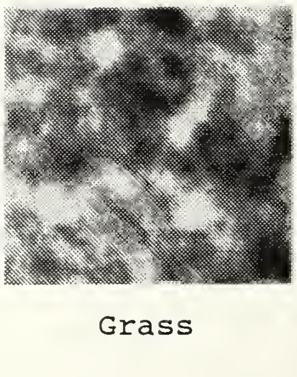
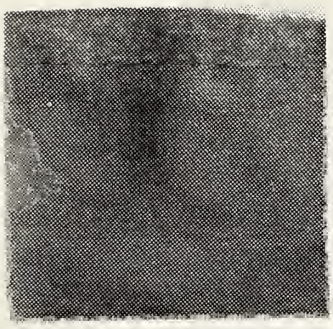
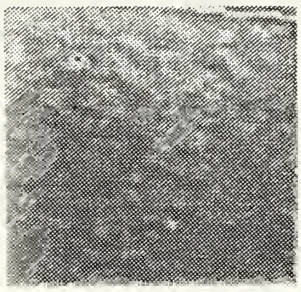

Grass

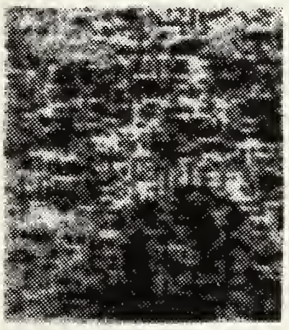
Figure 4-13 Image Textures Generated Using an ARIMA Model


Original Image

Contrast Enhanced

AR Generated
Contrast Enhanced
Image


Difference Operator
Result


AR Generated
Difference Operator
Result

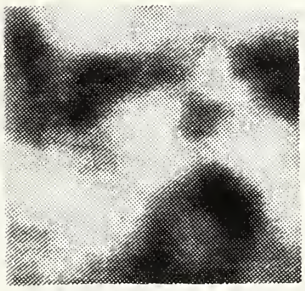

ARIMA Result

Figure 4-14 Final Test of ARIMA Model on Contrast Enhanced Trees (Magnification X2)

D. SUMMARY

The effectiveness of AR reproduction of image data using white noise input and filter coefficients calculated based on the statistics of the image signal is highly dependent on the nature of the image data. The water image, for example, with its smoothly varying and rather homogeneous nature, was quite well adapted to AR reproduction. Other images with more structure, abrupt variations, and more non-homogeneous characteristics, were not autoregressively reproducible to any great extent.

Using the ARIMA model, it seems that the operation of the inverse filter is very sensitive to the nature of the input data. Input data that are strictly based on the difference operator output can reproduce the original image, as was found in Chapter III. However, the AR model used to generate the inverse filter input (based on the statistics of image signal produced using the difference operator) does not generate image data accurately enough to reproduce images that resemble the real images tested.

V. CONCLUSIONS

This thesis sought to explore experimentally and to understand how linear filtering models could be used to generate texture in images. Of particular interest was the investigation of 2-D ARIMA models to see if they might be of any utility in this effort. Some time was spent exploring separable 2-D models to understand how transfer function pole placement affected image texture characteristics. Image textures generated using these models and applied to the summation filter yielded blurred or smooth textures with seemingly little variety or utility. The ultimate test of the model was its ability to reproduce actual image textures. The purely AR portion of the model reproduced a few types of actual textures well. However, the full ARIMA model failed to generate image textures that resembled the source images used. Many of the textures had strong edge differences that were not accurately reproducible by the AR model. Also, the summation filter developed seemed very sensitive to deviations in image signal data from that generated by application of the difference operator; that is, the procedure seemed not to be "robust."

Since many of the images tested here have definite edge structure, the difference image had lines which were not reproduced well by the AR model. Correspondingly the integrated AR model did not reproduce the original image. For this type of image, a combination of a line point process model [Ref. 11] with the integrator, would possibly have been more suitable. The image of trees had not such edge structure and produced somewhat better results. Further experimentation with images of this type and the ARIMA model would perhaps be appropriate.

APPENDIX A

COMPUTER PROGRAMS, SUBROUTINES, AND FUNCTIONS

Listed below are the names, associated computer systems, and functions of the various computer algorithms used to accomplish this thesis research. All programs were written by the thesis author unless otherwise noted. Program source codes are given at the end of this appendix (except for the MAKFIL* series).

A. PROGRAMS

1. AUTOREG (VAX/VMS FORTRAN)

The program did the following:

1) Generated a 128×128 zero mean white noise matrix using subroutine PGAUSS.

2) Multiplied the white noise by the appropriate image data standard deviation when necessary.

3) Converted that matrix into a displayable image file using subroutines SCALE and INTBYTE, when necessary.

4) Read filter parameters from an input file into an array.

5) Implemented the equation:

$$y(n,m) = -\sum_{\substack{i=0 \\ (i,j)\neq 0}}^{P-1} \sum_{j=0}^{Q-1} a_{ij} y(n-i,m-j) + w(n,m) \qquad (A.1)$$

using the white noise array and the filter coefficient array as inputs.

6) Converted the array result from 4) into a displayable image file using subroutines SCALE and INTBYTE.

7) Used subroutine SUBINTFILE to create image data files from filter results for further processing.

8) Used subroutine NONC to apply the summation filter to image data when necessary.

2. NONCAUSAL (VAX/VMS FORTRAN)

This program did the following:

1) Read filter coefficient values into an array.

2) Read image data from an input image file, converted it to integer values using subroutine BYTEINT, calculated the mean from the data, and placed the data into a real array.

3) Implemented the equation:

$$y(n,m) = \sum_{i=-L}^{L} \sum_{j=-L}^{L} a_{ij}x(n-i,m-j) \qquad (A.2)$$

using the image data array and the filter coefficient array.

4) Called the subroutine NONC to implement the equation in Step 3) a second time, when necessary.

5) Converted the result of Step 3) to a displayable image file using subroutine SCALE and INTBYTE.

3. CONV (VAX/VMX FORTRAN)

This program performs the same basic functions as NONCAUSAL, without having the capability of calling subroutine NONC. It was used for convenience in convolving certain filter structures with certain test images directly.

4. <u>MAKFIL* (VAX/VMS FORTRAN)</u>

This family of programs was used to create various autoregressive and FIR filter coefficient files, using source data manually entered into the program.

5. <u>SPECOR2 (IBM SYSTEM/370 3033 VS FORTRAN 1.4.1)</u>

This program implemented the equations derived in Appendix B and created data files used in developing the corresponding graphs.

6. <u>SPECOR3 (IBM SYSTEM/370 3033 VS FORTRAN 1.4.1)</u>

This program implemented the equations derived in Appendix D and created data files used in developing the corresponding graphs.

7. <u>SPECOR3A (IBM SYSTEM/370 3033 VS FORTRAN 1.4.1)</u>

This program implemented the equations derived in Appendix F and created data files used in developing corresponding graphs.

8. <u>VARIMGS (VAX/VMS FORTRAN)</u>

This program was used to display image data files on the COMTAL (not written by author).

9. <u>PIECE (VAX/VMS FORTRAN)</u>

This program was used to make 128×128 image data files from larger image data files.

10. <u>INTFILE (VAX/VMS FORTRAN)</u>

This program created appropriately formatted integer files from input image data for further processing.

11. <u>TRANS (VAX/VMS FORTRAN)</u>

This program changed the format of filter coefficient data files into a form readable by the image processing programs.

12. <u>NSHP (VAX/VMS FORTRAN)</u>

This program was used to convert quarter-plane autoregressive filter coefficient data to non-symmetric half-plane autoregressive filter coefficient data based on the transformation outlined in Chapter II, Section D.

B. SUBROUTINES

1. <u>PGAUSS (VAX/VMS FORTRAN)</u>

This subroutine, written by C.W. Therrien, was used to generate zero mean, unit variance white noise using RAN (a random number generator function) SQRT, COS, and SIN FORTRAN functions.

2. <u>SCALE (VAX/VMS FORTRAN)</u>

This subroutine takes an image data array and converts it to an integer array with values between an input maximum (MAX) and minimum (MIN) using the following scaling formula:

$$I(i,j) = \frac{(A(i,j)-LOW) \times (MAX-MIN)}{HIGH-LOW} + MIN \qquad (A.3)$$

A(i,j) is the input image data array, I(i,j) is the output integer array, and HIGH and LOW are the high and low values of A(i,j), respectively (calculated in this subroutine).

This is done to provide appropriate values for image files that will be displayed on the COMTAL Vision One/20, since the gray scale intensity level of each pixel is represented by an 8-bit word. So values possible (in base 10) range from 0 (darkest), to 255 (brightest).

3. INTBYTE AND BYTEINT (VAX/VMS FORTRAN)

These subroutines are necessary since data in an image file are stored in two's complement form. The related variable type in FORTRAN for these values is BYTE. To process image data using FORTRAN implementation of the appropriate formulas, these byte values must be converted to integer (and eventually real using the FLOAT function) form. Results of image processing formulas in real form must be converted to integer (using the INT function) and then byte form to be placed in image data files. INTBYTE converts integer type variables to byte type variables using the following criterion (I is an integer and B is a byte):

If $I \leq 127$ and $I \geq 0$ then $B = I$

If $I > 127$ and $I \leq 255$ then $B = I-256$

BYTEINT converts byte type variables to integer type variables using the following criterion:

If $B \geq -128$ and $B < 0$ then $I = B+256$

If $B \geq 0$ and $B \leq 127$ then $I = B$

4. SUBINTFILE (VAX/VMS FORTRAN)

This subroutine performed the same function as INTFILE, but could be called by a program to operate on processed image data arrays, rather than just image file data inputs.

5. NONC (VAX/VMX FORTRAN)

This subroutine performs essentially the same functions as NONCAUSAL, except that it can be called by a program to operate on an image data array.

C. APL FUNCTIONS

The APL systems on the IBM System/370 3033 and VAX/UNIX were used for matrix manipulations and operations, for graphing filter structures and convolution results, and for calculating autoregressive filter coefficients from image data. All APL functions except MAKMAT were written by C.W. Therrien.

1. MAKMAT (IBM)

This function was used to create the large coefficient matrices ($\bar{A}$) used in calculating the FIR filter coefficients as outlined in Chapter III.

2. CC2 (IBM)

This function was used to circularly convolve the Laplacian FIR filter and its various inverses. Appropriate zero-padding of these filters makes the resulting circular convolution equivalent to linear convolution [Ref. 2:pp. 70-72], which was the desired operation.

3. GETDATA (VAX/UNIX)

This function is used to transfer image data files from a UNIX subdirectory to an APL workspace.

4. PUTDATA (VAX/UNIX)

This function is used to transfer filter coefficient data files from an APL workspace to a UNIX subdirectory.

5. MEAN (VAX/UNIX)

This function is used to calculate the mean of an image data file for use in the APL function COVF.

6. COVF (VAX/UNIX)

This function is used to calculate terms in the 2-D covariance function for use in the APL function CORR.

7. CORR (VAX/UNIX)

This function is used to estimate the 2-D covariance function of the image data.

8. MVLEV (VAX/UNIX)

This function is used in APL function FF2DLEV to calculate necessary parameters for the 2-D Levinson recursion from the covariance function of the image data.

9. FF2DLEV (VAX/UNIX)

This function performs the 2-D Levinson recursion to solve for the filter coefficient vector.

```
c   THIS PROGRAM GENERATES AN IMAGE TEXTURE USING WHITE NOISE AS AN INPUT
c     TO AN AUTOREGRESSIVE FILTER WHOSE PARAMETERS ARE OBTAINED FROM THE
c     FILE FILCOEF. SUBROUTINE PGAUSS IS USED TO GENERATE THE INPUT WHITE
c     NOISE AND SUBROUTINES SCALE AND INTBYTE ARE USED TO PREPARE IMAGE
c     DATA ARRAYS FOR DISPLAY. SUBROUTINE SUBINTFILE IS USED IF AN
c     INTEGER FILE RESULT IS DESIRED, AND SUBROUTINE NONC IS USED IF A
c     LAPLACIAN INVERSE FILTERING STEP IS NEEDED.
c
c   DEFINE VARIABLES
        byte a(0:127),bim(0:127,0:127)
        integer n,seed,rsize,csize,pm1,qm1,i,j,jp1,row,col,integ(0:127,0:1
     ;27)
        real*8 val1,val2,wn(0:127,0:127),dsum,im(0:127,0:127),coef(0:9,0:9
     ;),var,max,min,imout(0:127,0:127)
c   OPEN FILES
        open(unit=1,name='[rathmann.data]wn.dat',type='new',access='direct
     ;',recordsize=32,maxrec=129)
        open(unit=2,name='[rathmann.im]final9a.dat',type='new',access='di
     ;rect',recordsize=32,maxrec=129)
        open(3,file='[rathmann.data]fcdbub.dat',status='old')
c   DEFINE PARAMETERS
        seed=1234567
        rsize=127
        csize=127
        pm1=3
        qm1=3
        max=229.0d0
        min=27.0d0
c   CREATE WHITE NOISE ARRAY
        do 10 i=0,rsize
          do 20 j=0,csize-1
            jp1=j+1
            call pgauss(seed,val1,val2)
            wn(i,j)=val1
            wn(i,jp1)=val2
   20     continue
   10 continue
c   SCALE ARRAY WN AND CONVERT TO BYTE FORM
c       call scale(wn,integ,max,min)
c       call intbyte(integ,bim)
c   WRITE THE WHITE NOISE IMAGE ARRAY TO A FILE
c       do 30 i=0,rsize
c         do 40 j=0,csize
c           a(j)=bim(i,j)
c  40     continue
c         write(1'i+1) (a(n),n=0,rsize)
c  30 continue
c   READ FILTER PARAMETERS INTO AN ARRAY
        do 50 i=0,pm1
          do 60 j=0,qm1
            read(3,55) coef(i,j)
   55       format(d20.12)
   60     continue
   50 continue
c   MULTIPLY WHITE NOISE BY REAL IMAGE STANDARD DEVIATION
        var=sqrt(coef(0,0))
        do 70 i=0,rsize
          do 80 j=0,csize
            wn(i,j)=wn(i,j)*var
   80     continue
   70 continue
 c  APPLY WHITE NOISE TO THE AUTOREGRESSIVE FILTER
       do 110 n=0,rsize
         do 120 m=0,csize
           dsum=0.0d0
           do 130 i=0,pm1
             do 140 j=0,pm1
               if((i.eq.0).and.(j.eq.0)) go to 140
               row=n-i
               col=m-j
               if((row.lt.0).or.(col.lt.0)) go to 140
               dsum=(coef(i,j)*in(row,col))+dsum
  140        continue
  130      continue
           in(n,m)=(-1.0d0*dsum)+wn(n,m)
  120    continue
  110 continue
 c  FILTER THE IMAGE DATA ARRAY USING THE LAPLACIAN INVERSE FILTER
       call nonc(in,imout)
 c  SCALE THE RESULTING IMAGE ARRAY AND CONVERT TO BYTE FORM
       call scale(imout,integ,max,min)
       call intbyte(integ,bim)
 c  WRITE THE GENERATED IMAGE INTO A FILE
       do 150 i=0,rsize
         do 160 j=0,csize
           a(j)=bim(i,j)
  160    continue
         write(2'i+1) (a(n),n=0,rsize)
  150 continue
 c  CLOSE FILES
       close(unit=1)
       close(unit=2)
       close(3)
       stop
       end
```

NONCAUSAL

```
c   THIS PROGRAM GENERATES AN IMAGE TEXTURE USING A NONCAUSAL FIR FILTER
c      WHOSE PARAMETERS ARE OBTAINED FROM A DATA FILE. THE FILTER IS
c      APPLIED TO AN IMAGE. SUBROUTINES SCALE, INTBYTE, AND BYTEINT ARE
c      USED TO PREPARE IMAGE DATA ARRAYS FOR DISPLAY. SUBROUTINE NONC
c      IS USED IF AN INTERMEDIATE FILTERING STEP IS DESIRED.
c
c   DEFINE VARIABLES
      byte a(0:127),bim(0:127,0:127)
      integer n,rsize,csize,i,j,row,col,integ(0:127,0:127),fsize,mfsize,
     *index1,index2
      real*8 dsum,in(0:127,0:127),coef(-10:10,-10:10),msum,mean,dump(0:1
     !27,0:127),high,low,trv(-1:128,-1:128),max,min
c   OPEN FILES
      open(unit=1,name='[rathmann.im]park.dat',type='old',access='dir
     !ect',recordsize=32,maxrec=128)
c      open(unit=2,name='[rathmann.im]arim6A.dat',type='new',access='dire
c     !ct',recordsize=32,maxrec=128)
      open(3,file='[rathmann.data]ncfilco3a.dat',status='old')
```

```
c   DEFINE PARAMETERS
        index1=-1
        index2=128
        rsize=127
        csize=127
        fsize=1
        mfsize=-1*fsize
        max=255.0d0
        min=0.0d0
c   READ FILTER PARAMETERS INTO AN ARRAY
        do 10 i=mfsize,fsize
          do 20 j=mfsize,fsize
            read(3,25) coef(i,j)
  25        format(d20.12)
  20      continue
  10    continue
c   READ IMAGE TO BE FILTERED INTO AN ARRAY AND CONVERT TO INTEGER
        do 30 i=0,rsize
          read(1'i+1) (a(n),n=0,127)
          do 40 j=0,csize
          bim(i,j)=a(j)
  40      continue
  30    continue
        call byteint(bim,integ)
c   CONVERT THE INTEGER ARRAY INTO A REAL ARRAY AND COMPUTE THE MEAN
        msum=0.0d0
        high=-10000.0d0
        low=10000.0d0
        do 50 i=0,rsize
          do 60 j=0,csize
            im(i,j)=float(integ(i,j))
            msum=im(i,j)+msum
            if(im(i,j).gt.high) high=im(i,j)
            if(im(i,j).lt.low) low=im(i,j)
  60      continue
  50    continue
        mean=msum/(128.0*128.0)
        write(*,55)mean,high,low
  55 format(' original image data- mean=',d12.5,' high=',d12.5,' low='d
    +12.5)
c   SUBTRACT THE MEAN FROM THE DATA
        do 70 i=0,127
          do 80 j=0,127
            im(i,j)=im(i,j)-mean
  80      continue
  70    continue
c   FILTER THE IMAGE ARRAY
        do 90 n=index1,index2
          do 100 m=index1,index2
            dsum=0.0d0
            do 110 i=mfsize,fsize
              do 120 j=mfsize,fsize
                row=n-i
                col=m-j
                if((row.lt.0).or.(col.lt.0)) go to 120
                if((row.gt.127).or.(col.gt.127)) go to 120
                dsum=(im(row,col)*coef(i,j))+dsum
 120          continue
 110        continue
            try(n,m)=dsum
 100      continue
  90    continue
c   CALL SUBROUTINE TO INVERSE FILTER FILTERED IMAGE ARRAY
c        call nonc(try,im)
```

```
c  ADD THE MEAN OF THE INPUT IMAGE TO THE FILTERED RESULT
      do 145 i=0,rsize
        do 155 j=0,csize
          im(i,j)=try(i,j)+mean
 155    continue
 145  continue
c  CREATE AN IMAGE DATA FILE
      call supintfile(im)
c  SCALE THE RESULTING IMAGE ARRAY AND CONVERT TO BYTE FORM
      call scale(im,integ,max,min)
      call intbyte(integ,bim)
c  WRITE THE GENERATED IMAGE INTO A FILE
      do 150 i=0,rsize
        do 160 j=0,csize
          a(j)=bim(i,j)
 160    continue
        write(2'i+1) (a(n),n=0,rsize)
 150  continue
c  CLOSE FILES
      close(unit=1)
      close(unit=2)
      close(3)
      stop
      end
```

CONV

```
c  THIS PROGRAM GENERATES AN IMAGE TEXTURE USING A NONCAUSAL FIR FILTER
c    WHOSE PARAMETERS ARE OBTAINED FROM A DATA FILE. THE FILTER IS
c    APPLIED TO AN IMAGE. SUBROUTINES SCALE, INTBYTE, AND BYTEINT ARE
c    USED TO PREPARE IMAGE DATA ARRAYS FOR DISPLAY.
c
c  DEFINE VARIABLES
      byte a(0:127),bim(0:127,0:127)
      integer n,rsize,csize,i,j,row,col,integ(0:127,0:127),fsize,mfsize
      real*8 dsum,im(0:127,0:127),coef(-11:11,-11:11),msum,mean,dump(0:1
     :27,0:127)
c  OPEN FILES
      open(unit=1,name='[rathmann.im]arim53.dat',type='old',access='dire
     :ct',recordsize=32,maxrec=128)
      open(unit=2,name='[rathmann.im]arim59.dat',type='new',access='dire
     :ct',recordsize=32,maxrec=128)
      open(3,file='[rathmann.data]conv21.dat',status='old')
c  DEFINE PARAMETERS
      rsize=127
      csize=127
      fsize=11
      mfsize=-1*fsize
c  READ FILTER PARAMETERS INTO AN ARRAY
      do 10 i=mfsize,fsize
        do 20 j=mfsize,fsize
          read(3,25) coef(i,j)
 25       format(d20.12)
 20     continue
 10   continue
c  READ IMAGE TO BE FILTERED INTO AN ARRAY AND CONVERT TO INTEGER
      do 30 i=0,rsize
        read(1'i+1) (a(n),n=0,127)
        do 40 j=0,csize
        bim(i,j)=a(j)
 40     continue
 30   continue
      call byteint(bim,integ)
```

```
c  CONVERT THE INTEGER ARRAY INTO A REAL ARRAY AND COMPUTE THE MEAN
      msum=0.0d0
      do 50 i=0,rsize
        do 60 j=0,csize
          in(i,j)=float(integ(i,j))
          msum=in(i,j)+msum
   60   continue
   50 continue
      mean=msum/float((rsize+1)*(csize+1))
      write(*,55)mean
   55 format(f20.12)
c  SUBTRACT THE MEAN FROM THE IMAGE ARRAY
      do 70 i=0,rsize
        do 80 j=0,csize
          in(i,j)=in(i,j)-mean
   80   continue
   70 continue
c  FILTER THE IMAGE ARRAY
      do 90 n=0,rsize
        do 100 m=0,csize
          dsum=0.0d0
          do 110 i=-fsize,fsize
            do 120 j=-fsize,fsize
              row=n-i
              col=m-j
              if((row.lt.0).or.(col.lt.0)) go to 120
              if((row.gt.127).or.(col.gt.127)) go to 120
              dsum=(in(row,col)*coef(i,j))+dsum
  120       continue
  110     continue
          dump(n,m)=dsum
  100   continue
   90 continue
c  SCALE THE RESULTING IMAGE ARRAY AND CONVERT TO BYTE FORM
      call scale(dump,integ)
      call intbyte(integ,bim)
c  WRITE THE GENERATED IMAGE INTO A FILE
      do 150 i=0,rsize
        do 160 j=0,csize
          a(j)=bim(i,j)
  160   continue
        write(2'i+1) (a(n),n=0,rsize)
  150 continue
c  CLOSE FILES
      close(unit=1)
      close(unit=2)
      close(3)
      stop
      end
```

SPECOR2

```
C  THIS PROGRAM SCLVES EQUATIONS FOR SPECTRAL CONTENT AND CORRELATION
C    IN ONE DIRECTION OF A GIVEN AUTOREGRESSIVE IMAGE MODEL.  IT WRITES
C    THESE RESULTS TO DEVICES 3 AND 4 RESPECTIVELY.
C  DEFINE VARIABLES
      INTEGER I,K
      REAL*8 PI,ALPHA,THETA,W,ALPHAS,SXW,RXK(-49:49),Z,ALPHAK
C  DEFINE PARAMETERS
      PI=3.141592654
      ALPHA=0.999999
      ALPHAS=ALPHA**2
C  START X AXIS LOOP AND DEFINE X VALUES
      DO 10 I=0,99
        W=(-1.0*PI)+(2.0*PI*(FLOAT(I)/99.0))
C  DO SPECTRAL ANALYSIS
        SXW=1.0/(1.0-(2.0*ALPHA*COS(W))+ALPHAS)
        WRITE(3,15)W,SXW
   15   FORMAT(F10.5,1X,F10.5)
   10 CONTINUE
C  DC CORRELATION FUNCTION SOLUTION
      DO 20 K=0,49
        ALPHAK=ALPHA**K
        RXK(K)=(ALPHAK/(1.0-ALPHAS))
        I=-1*K
        RXK(I)=RXK(K)
   20 CONTINUE
      DO 30 K=-49,49
        Z=FLOAT(K)
        WRITE(4,25)Z,RXK(K)
   25   FORMAT(F10.0,1X,F30.15)
   30 CONTINUE
      STOP
      END
```

```
C  THIS PROGRAM SOLVES EQUATIONS FOR SPECTRAL CONTENT AND CORRELATION
C     IN ONE DIRECTION OF A GIVEN AUTOREGRESSIVE IMAGE MODEL.  IT WRITES
C     THESE RESULTS TO DEVICES 3 AND 4 RESPECTIVELY.
C  DEFINE VARIABLES
      INTEGER I,J,M
      REAL*8 PI,ALPHA,THETA,ALPHA3,AL3PA,ALPHAS,ALPHA4,TWOTH,COS2T,W,A,B
     *,COS2W,SXW1,SXW,K,ALPHAK,KTHETA,ALPH2K,KP2TMP,KTMP,INTERM,RXK(-49:
     *49),Z
C  DEFINE PARAMETERS
      PI=3.141592654
      ALPHA=0.9
      THETA=0.0*(PI/12.0)
      IF(THETA.EQ.0.0) THETA=0.000001
      ALPHA3=ALPHA**3
      AL3PA=ALPHA3+ALPHA
      ALPHAS=ALPHA**2
      ALPHA4=ALPHA**4
      COS2T=COS(2.0*THETA)
      TWOTH=2.0*THETA
C  START X AXIS LOOP AND DEFINE X VALUES
C     DO 10 I=0,99
C       W=(-1.0*PI)+(2.0*PI*(FLOAT(I)/99.0))
C  DO SPECTRAL ANALYSIS
C       A=THETA-W
C       B=THETA+W
C       COS2W=COS(2.0*W)
C       SXW1=ALPHAS-(COS(A)*AL3PA)+(ALPHAS*COS2T)-(COS(B)*AL3PA)+(ALPHAS
C    **COS2W)
C       SXW=1.0/(1.0+ALPHA4+(2.0*SXW1))
C       WRITE(3,15)W,SXW
C  15   FORMAT(F10.5,1X,F10.5)
C  10 CONTINUE
C  DO CORRELATION FUNCTION SOLUTION
        DO 30 J=0,49
          K=FLOAT(J)
          ALPHAK=ALPHA**J
          KTHETA=K*THETA
          KP2TMP=((2.0+K)*THETA)-PI
          KTMP=KTHETA-PI
          INTERM=(COS(KP2TMP)-(ALPHAS*COS(KTMP)))/(1.0+ALPHA4-(2.0*ALPHA
     *S*COS(TWOTH)))
          RXK(J)=(ALPHAK/(2.0*(SIN(THETA)**2)))*((COS(KTHETA)/(1.0-ALPHA
     *S))+INTERM)
          GO TO 36
C  35     RXK(J)=(ALPHAK*(2.0+K))/(2.0*((1.0-ALPHAS)**2))
   36     M=-1*J
          RXK(M)=RXK(J)
   30 CONTINUE
      DO 40 J=-49,49
        Z=FLOAT(J)
        WRITE(4,45)Z,RXK(J)
   45   FORMAT(F10.1,1X,F30.15)
   40 CONTINUE
      STOP
      END
```

SPECOR3A

```
C  THIS PROGRAM SOLVES EQUATIONS FOR SPECTRAL CONTENT AND CORRELATION
C     IN ONE DIRECTION OF A GIVEN AUTOREGRESSIVE IMAGE MODEL.  IT WRITES
C     THESE RESULTS TO DEVICES 3 AND 4 RESPECTIVELY.
C  DEFINE VARIABLES
      INTEGER I,J,M
      REAL*8 PI,A,B,AB,AMBS,AS,BS,AK1,AK2,BK1,BK2,INTERM,RXK(-49:49),Z,A
     *K,KP2,KP1,OPAS,OMAS
C  DEFINE PARAMETERS
      PI=3.141592654
      A=-0.9
      B=-0.89999
      AB=A*B
      AMBS=(A-B)**2
      AS=A**2
      BS=B**2
C  START X AXIS LOOP AND DEFINE X VALUES
      DO 10 I=0,99
        W=(-1.0*PI)+(2.0*PI*(FLOAT(I)/99.0))
C  DO SPECTRAL ANALYSIS
        COS2W=COS(2.0*W)
        SXW1=1.0-(2.0*(A+B+(AS*B)+(A*BS))*COS(W))+(2.0*A*B*COS2W)
        SXW=1.0/(SXW1+AS+(2.0*A*B)+BS+(AS*BS))
        WRITE(3,15)W,SXW
 15     FORMAT(F10.5,1X,F10.5)
 10 CONTINUE
C  DO CORRELATION FUNCTION SOLUTION
        DO 30 J=0,49
          IF(A.EQ.B) GO TO 35
          AK1=A**(J+1)
          AK2=A**(J+2)
          BK1=B**(J+1)
          BK2=B**(J+2)
          INTERM=(AK2/(1.0-AS)-(((AK1*B)+A*BK1))/(1.0-AB))+(BK2/(1.0-BS)
     *)
          RXK(J)=(1.0/AMBS)*INTERM
          GO TO 36
 35       AK=A**J
          KP2=FLOAT(J+2)
          KP1=FLOAT(J+1)
          OPAS=1.0+AS
          OMAS=1.0-AS
          RXK(J)=((KP2*KP1*AK)/(2.0*OMAS))-((FLOAT(J)*KP1*AK)/(2.0*OPAS)
     *)
 36       M=-1*J
          RXK(M)=RXK(J)
 30 CONTINUE
    DO 40 J=-49,49
      Z=FLOAT(J)
      WRITE(4,45)Z,RXK(J)
 45   FORMAT(F10.1,1X,F30.15)
 40 CONTINUE
    STOP
    END
```

PIECE

```
c   THIS PROGRAM TAKES A 128X128 PIECE OUT OF A 512X512 IMAGE DATA FILE
c       AND PLACES THAT DATA INTO AN IMAGE FILE
c
c
c
c
c   DEFINE VARIABLES
        byte a(0:511),bim(0:511,0:511),ao(0:127)
c   OPEN FILES
        open(unit=1,name='[rathmann.im]trees.dat',type='old',access='dire
       1ct',recordsize=32,maxrec=128)
        open(unit=2,name='[rathmann.im]trees64.dat',type='new',access='dir
       1ect',recordsize=16,maxrec=64)
c   READ IMAGE TO BE FILTERED INTO AN ARRAY
        do 30 i=0,127
          read(1'i+1) (a(n),n=0,127)
          do 40 j=0,127
          bim(i,j)=a(j)
   40     continue
   30 continue
c   WRITE THE DESIRED IMAGE PORTION INTO A FILE
        do 150 i=63,126
          do 160 j=63,126
            k=j-63
            ao(k)=bim(i,j)
  160     continue
          write(2'i-62) (ao(n),n=0,63)
  150 continue
c   CLOSE FILES
        close(unit=1)
        close(unit=2)
        stop
        end
```

INTFILE

```
c   THIS PROGRAM CREATES A 128X128 OR 64X64 INTEGER FILE FROM A 128X128
c     INPUT IMAGE FILE
c
c
c
c
c   DEFINE VARIABLES
        byte a(0:127),bim(0:127,0:127)
        integer i,j,n,k,l,integ(0:127,0:127),sum,high,low
        real*8 mean
c   OPEN FILES
        open(unit=1,name='[rathmann.im]difar2.dat',type='old',access='d
       1irect',recordsize=32,maxrec=128)
c         open(2,file='[rathmann.data]dinwat.dat',status='new')
c   READ IMAGE TO BE FILTERED INTO AN ARRAY
        do 30 i=0,127
          read(1'i+1) (a(n),n=0,127)
          do 40 j=0,127
          bim(i,j)=a(j)
   40     continue
   30 continue
c   CONVERT THE BYTE ARRAY INTO AN INTEGER ARRAY
        call byteint(bim,integ)
```

INTFILE (CONT.)

```
c  COMPUTE HIGH AND LOW VALUES OF THE IMAGE DATA AND WRITE TO TERMINAL
      high=0
      low=255
      do 31 i=0,127
        do 32 j=0,127
          if(integ(i,j).lt.low) low=integ(i,j)
          if(integ(i,j).gt.high) high=integ(i,j)
   32   continue
   31 continue
      write(*,33)high,low
   33 format(2i5)
c  COMPUTE MEAN OF INTEGER ARRAY AND SUBTRACT IT FROM THE DATA
c       sum=0
c       do 35 i=0,127
c         do 45 j=0,127
c           sum=sum+integ(i,j)
c   45    continue
c   35 continue
c      mean=float(sum)/(128.0*128.0)
c       write(*,34)sum,mean
c   34 format(i10,e10.3)
c       do 35 i=0,127
c         do 45 j=0,127
c           integ(i,j)=integ(i,j)-int(mean+0.5)
c   45    continue
c   35 continue
c  WRITE INTEGER ARRAY INTO A 128x128 DATA FILE
c       k=128
c       l=128
c       write(2,39)k,l
c   39 format(2i5)
c       do 50 i=0,127
c         do 60 j=0,112,16
c           write(2,70)integ(i,j),integ(i,j+1),integ(i,j+2),integ(i,j+3),i
c     *nteg(i,j+4),integ(i,j+5),integ(i,j+6),integ(i,j+7),integ(i,j+8),in
c     *teg(i,j+9),integ(i,j+10),integ(i,j+11),integ(i,j+12),integ(i,j+13)
c     *,integ(i,j+14),integ(i,j+15)
c   70      format(16i5)
c   60    continue
c   50 continue
c  CLOSE FILES
      close(unit=1)
      close(unit=2)
      stop
      end
```

TRANS

```
c  THIS PROGRAM READS DATA OUT OF A FILE IN FREE FORMAT AND CONVERTS
c    IT TO A PROGRAM READABLE FORMAT.
      integer i,j
      real*8 a(16)
      open(1,file='[rathmann.data]adwat.dat',status='old')
      open(2,file='[rathmann.data]fdwat.dat',status='new')
      read(1,*)i,j,a(1),a(2),a(3),a(4),a(5),a(6),a(7),a(8),a(9),a(10),a(
     *11),a(12),a(13),a(14),a(15),a(16)
      do 10 k=1,16
        write(2,20)a(k)
   20   format(d20.12)
   10 continue
      stop
      end
```

NSHP

```
c   THIS PROGRAM GENERATES AN IMAGE TEXTURE USING WHITE NOISE AS AN INPUT
c     TO AN AUTOREGRESSIVE FILTER WHOSE PARAMETERS ARE OBTAINED FROM THE
c     FILE FILCOEF. SUBROUTINE PGAUSS IS USED TO GENERATE THE INPUT WHITE
c     NOISE AND SUBROUTINES SCALE AND INTBYTE ARE USED TO PREPARE IMAGE
c     DATA ARRAYS FOR DISPLAY.
c
c   DEFINE VARIABLES
        byte a(0:127),bim(0:127,0:127)
        integer n,seed,rsize,csize,pm1,qm1,i,j,jp1,row,col,integ(0:127,0:1
       :27),mpm1
        real*8 val1,val2,wn(0:127,0:127),dsum,im(0:127,0:127),coef(-2:9,-2
       ::9)
c   OPEN FILES
        open(unit=1,name='[rathmann.data]wn.dat',type='new',access='direct
       :',recordsize=32,maxrec=128)
        open(unit=2,name='[rathmann.data]rotate.dat',type='new',access='di
       :rect',recordsize=32,maxrec=128)
        open(3,file='[rathmann.data]fc3a.dat',status='old')
c       open(4,file='[rathmann.data]ts.dat',status='new')
c   DEFINE PARAMETERS
        seed=1234567
        rsize=127
        csize=127
        pm1=2
        qm1=4
        mpm1=-1*pm1
c   CREATE WHITE NOISE ARRAY
        do 10 i=0,rsize
          do 20 j=0,csize-1
            jp1=j+1
            call pgauss(seed,val1,val2)
            wn(i,j)=val1
            wn(i,jp1)=val2
   20     continue
   10 continue
c   SCALE ARRAY WN AND CONVERT TO BYTE FORM
        call scale(wn,integ)
        call intbyte(integ,bim)
c   WRITE THE WHITE NOISE IMAGE ARRAY TO A FILE
        do 30 i=0,rsize
          do 40 j=0,csize
            a(j)=bim(i,j)
   40     continue
          write(1'i+1) (a(n),n=0,rsize)
   30 continue
c   READ FILTER PARAMETERS INTO AN ARRAY
        do 50 i=mpm1,pm1
          do 60 j=0,qm1
            read(3,55) coef(i,j)
   55       format(d20.12)
   60     continue
   50 continue
c   APPLY WHITE NOISE TO THE AUTOREGRESSIVE FILTER
        do 110 m=0,csize
          do 120 n=0,rsize
            dsum=0.0d0
            do 130 i=mpm1,pm1
              do 140 j=0,qm1
                if((i.eq.0).and.(j.eq.0)) go to 140
                row=n-i
                col=m-j
                if((row.lt.0).or.(col.lt.0)) go to 140
                dsum=(coef(i,j)*im(row,col))+dsum
  140         continue
  130       continue
            im(n,m)=(-1.0d0*dsum)+wn(n,m)
  120     continue
  110 continue
```

NSHP (CONT.)

```
c  SCALE THE RESULTING IMAGE ARRAY AND CONVERT TO BYTE FORM
      call scale(im,integ)
      call intbyte(integ,bim)
c  WRITE THE GENERATED IMAGE INTO A FILE
      do 150 i=0,rsize
        do 160 j=0,csize
          a(j)=bim(i,j)
 160    continue
        write(2'i+1) (a(n),n=0,rsize)
 150  continue
c  CLOSE FILES
      close(unit=1)
      close(unit=2)
      close(3)
c     close(4)
      stop
      end
```

PGAUSS

```
      SUBROUTINE PGAUSS(K,Z1,Z2)
      REAL*8 A,B,Z1,Z2
      INTEGER*4 K
      A = SQRT(-2.0D0 * ALOG(RAN(K)))
      B = 6.2831853D0 * RAN(K)
      Z1 = A * COS(B)
      Z2 = A * SIN(B)
      RETURN
      END
```

SCALE

```
      subroutine scale(arr,integ,max,min)
c  THIS SUBROUTINE SCALES AN ARRAY TO INTEGER VALUES BETWEEN A GIVEN
c    MAXIMUM AND MINIMUM
c  DEFINE VARIABLES
      integer integ(0:127,0:127),i,j
      real*8 arr(0:127,0:127),high,low,mean,max,min
c  PERFORM SCALING
      high=-10000.0d0
      low=10000.0d0
      do 10 i=0,127
        do 20 j=0,127
          if(arr(i,j).le.low) low=arr(i,j)
          if(arr(i,j).ge.high) high=arr(i,j)
          sum=arr(i,j)+sum
 20     continue
 10   continue
      mean=sum/(128.0*128.0)
      write(*,25)high,low,mean
 25   format(' prescaled image data- high=',d12.5,' low=',d12.5,' mean='
     !,d12.5)
c  CONTINUE SCALING AND CONVERT TO INTEGER FORM
      do 30 i=0,127
        do 40 j=0,127
          integ(i,j)=int(((arr(i,j)-low)*((max-min)/(high-low)))+0.5d0)
          integ(i,j)=integ(i,j)+int(min)
 40     continue
 30   continue
```

```
c   CALCULATE HIGH, LOW, AND MEAN OF SCALED IMAGE
        sum=0.0d0
        high=-10000.0d0
        low=10000.0d0
        do 50 i=0,127
          do 60 j=0,127
            if(float(integ(i,j)).gt.high) high=float(integ(i,j))
            if(float(integ(i,j)).lt.low) low=float(integ(i,j))
            sum=float(integ(i,j))+sum
   60     continue
   50 continue
      mean=sum/(128.0*128.0)
      write(*,65)high,low,mean
   65 format(' scaled image data- high=',d12.5,' low=',d12.5,' mean=',d1
     :2.5)
      return
      end
```

INTBYTE

```
      subroutine intbyte(integ,bim)
c   THIS SUBROUTINE TAKES AN INTEGER ARRAY AND CONVERTS IT INTO A
c     BYTE ARRAY
c   DEFINE VARIABLES
      integer i,j,integ(0:127,0:127),n
      byte bim(0:127,0:127)
c   PERFORM CONVERSION
      n=0
      do 10 i=0,127
        do 20 j=0,127
          if((integ(i,j).lt.0).or.(integ(i,j).gt.255)) n=n+1
          if(integ(i,j).lt.0) bim(i,j)=-128
          if(integ(i,j).gt.255) bim(i,j)=127
          if((integ(i,j).le.127).and.(integ(i,j).ge.0)) bim(i,j)=
     :integ(i,j)
          if((integ(i,j).gt.127).and.(integ(i,j).le.255)) bim(i,j)=
     :integ(i,j)-256
   20   continue
   10 continue
      write(*,30) n
   30 format(' THE NUMBER OF POINTS OUT OF RANGE IS',i5)
      return
      end
```

BYTEINT

```
      subroutine byteint(bim,integ)
c  THIS PROGRAM TAKES A BYTE ARRAY AND CONVERTS IT INTO AN INTEGER ARRAY
c
c  DEFINE VARIABLES
      integer integ(0:127,0:127),i,j,n
      byte bim(0:127,0:127)
c  PERFORM CONVERSION
      n=0
      do 10 i=0,127
        do 20 j=0,127
          if((bim(i,j).lt.-128).or.(bim(i,j).gt.127)) n=n+1
          if(bim(i,j).lt.-128) integ(i,j)=-128
          if(bim(i,j).gt.127) integ(i,j)=127
          if((bim(i,j).ge.-128).and.(bim(i,j).lt.0)) integ(i,j)=
     1bim(i,j)+256
          if((bim(i,j).ge.0).and.(bim(i,j).le.127)) integ(i,j)=bim(i,j)
 20     continue
 10   continue
      write(*,30)n
 30   format(' THE NUMBER OF POINTS OUT OF RANGE IS',I5)
      return
      end
```

SUBINTFILE

```
      subroutine subintfile(im)
c  THIS SUBROUTINE CREATES A 128x128 INTEGER DATA FILE FROM FILTERED
c    IMAGE DATA
c
c  DEFINE VARIABLES
      integer i,j,n,k,l,integ(0:127,0:127),sum
      real*8 mean,im(0:127,0:127),high,low
c  OPEN FILES
      open(1,file='[rathmann.data]dinpar.dat',status='new')
c  COMPUTE HIGH AND LOW VALUES OF THE IMAGE DATA AND WRITE TO TERMINAL
      high=-1.0d5
      low=1.0d5
      do 31 i=0,127
        do 32 j=0,127
          if(im(i,j).lt.low) low=im(i,j)
          if(im(i,j).gt.high) high=im(i,j)
 32     continue
 31   continue
      write(*,33)high,low
 33   format(2d20.12)
c  COMPUTE MEAN OF INTEGER ARRAY AND SUBTRACT IT FROM THE DATA
c      sum=0
c      do 35 i=0,127
c        do 45 j=0,127
c          sum=sum+integ(i,j)
c 45     continue
c 35   continue
c     mean=float(sum)/(128.0*128.0)
c      write(*,34)sum,mean
c 34   format(i10,e10.3)
c      do 35 i=0,127
c        do 45 j=0,127
c          integ(i,j)=integ(i,j)-int(mean+0.5)
c 45     continue
c 35   continue
```

SUBINTFILE

```
c   CONVERT IMAGE DATA INTO INTEGER FORM
        do 45 i=0,127
          do 55 j=0,127
            integ(i,j)=int(im(i,j)+0.5d0)
    55    continue
    45 continue
c   WRITE INTEGER ARRAY INTO A 128X128 DATA FILE
        k=128
        l=128
        write(1,39)k,l
    39 format(2i5)
        do 50 i=0,127
          do 60 j=0,112,16
            write(1,70)integ(i,j),integ(i,j+1),integ(i,j+2),integ(i,j+3),i
       *nteg(i,j+4),integ(i,j+5),integ(i,j+6),integ(i,j+7),integ(i,j+8),in
       *teg(i,j+9),integ(i,j+10),integ(i,j+11),integ(i,j+12),integ(i,j+13)
       *,integ(i,j+14),integ(i,j+15)
    70      format(16i5)
    60    continue
    50 continue
c   CLOSE FILES
        close(unit=1)
        return
        end
```

NONC

```
        subroutine nonc(try,im)
c   THIS SUBROUTINE TAKES AN IMAGE ARRAY AND FILTERS IT WITH A
c     NONCAUSAL FIR FILTER.
c
c   DEFINE VARIABLES
        integer n,rsize,csize,i,j,row,col,fsize,mfsize,index1,index2
        real*8 dsum,im(0:127,0:127),coef(-10:10,-10:10),msum,mean,dump(-1:
       :128,-1:128),high,low,try(0:127,0:127)
c   OPEN FILES
        open(1,file='[rathmann.data]ncfilcp21.dat',status='old')
c   DEFINE PARAMETERS
        index1=0
        index2=127
        rsize=127
        csize=127
        fsize=10
        mfsize=-1*fsize
c   READ FILTER PARAMETERS INTO AN ARRAY
        do 10 i=mfsize,fsize
          do 20 j=mfsize,fsize
            read(1,25) coef(i,j)
    25      format(d20.12)
    20    continue
    10 continue
c   TAKE THE INPUT ARRAY AND COMPUTE THE MEAN, HIGH, AND LOW VALUES
        msum=0.0d0
        high=-10000.0d0
        low=10000.0d0
        do 50 i=index1,index2
          do 60 j=index1,index2
            msum=try(i,j)+msum
            if(try(i,j).gt.high) high=try(i,j)
            if(try(i,j).lt.low) low=try(i,j)
    60    continue
    50 continue
        mean=msum/float((index2-index1+1)*(index2-index1+1))
        write(*,55)mean,high,low
    55 format(' input image array- mean=',d12.5,' high=',d12.5,' low=',d1
       *2.5)
```

```
c  FILTER THE IMAGE ARRAY
        do 90 n=index1,index2
          do 100 m=index1,index2
            dsum=0.0d0
            do 110 i=-fsize,fsize
              do 120 j=-fsize,fsize
                row=n-i
                col=m-j
                if((row.lt.index1).or.(col.lt.index1)) go to 120
                if((row.gt.index2).or.(col.gt.index2)) go to 120
                dsum=(try(row,col)*coef(i,j))+dsum
  120         continue
  110       continue
            im(n,m)=dsum
  100     continue
   90 continue
c  CONVERT THE 130X130 ARRAY INTO A 128X128 ARRAY, IF NECESSARY
c       do 130 i=0,127
c         do 140 j=0,127
c           im(i,j)=dump(i,j)
c  140    continue
c  130 continue
        close(1)
        return
        end
```

```
     ∇ R←MAKMAT;⎕IO;A;C;D;E;F;G;H;I;J;K;L;M;O
      ⎕IO←0
      A←11 11 66 ⍴0
      A[0;0;0]←4
      A[0;0;1]←¯4
      DUM←0
      I←0
      START1←0
      START2←¯1
    LOOP1:I←I+1
      START1←START1+(I-1)
      START2←START2+(I+1)
      ROW←¯1
    LOOP2:ROW←ROW+1
      →(ROW=0)/ROW1
      →(ROW=I)/ROWI
      COL1←COL1+1
      COL2←COL1+(I-1)
      COL3←COL2+1
      COL4←COL3+1
      COL5←COL4+I
      A[I;ROW;COL1]←¯1
      A[I;ROW;COL2]←¯1
      A[I;ROW;COL3]←4
      A[I;ROW;COL4]←¯1
      →(I=10)/LOOP2
      A[I;ROW;COL5]←¯1
      →(ROW<I)/LOOP2
    ROW1:COL1←START1
      COL2←COL1+I
      COL3←COL2+1
      COL4←COL3+I
      A[I;ROW;COL1]←¯1
      A[I;ROW;COL2]←4
      A[I;ROW;COL3]←¯2
      →(I=10)/LOOP2
      A[I;ROW;COL4]←¯1
      →(DUM=0)/LOOP2
    ROWI:COL1←START2
      COL2←COL1+1
      COL3←COL2+(I+1)
      A[I;ROW;COL1]←¯2
      A[I;ROW;COL2]←4
      →(I=10)/DONE
      A[I;ROW;COL3]←¯2
    DONE:→(I<9)/LOOP1
      C←¯10 0 ↓A[0;;]
      D←¯9 0 ↓A[1;;]
      E←¯8 0 ↓A[2;;]
      F←¯7 0 ↓A[3;;]
      G←¯6 0 ↓A[4;;]
      H←¯5 0 ↓A[5;;]
      K←¯4 0 ↓A[6;;]
      L←¯3 0 ↓A[7;;]
      M←¯2 0 ↓A[8;;]
      N←¯1 0 ↓A[9;;]
      O←A[10;;]
      C←C,[0] D
      D←C,[0] E
      E←D,[0] F
      F←E,[0] G
      G←F,[0] H
      H←G,[0] K
      K←H,[0] L
      L←K,[0] M
      M←L,[0] N
      R←M,[0] O
```

CC2

```
     ∇ Y←X CC2 H;I1;I2;⎕IO;N1;N2;T1;T2
      ⍝
      ⍝ FUNCTION TO DO 2-D CIRCULAR CONVOLUTION
      ⍝
       →((∧/(⍴X)=⍴H)∧2=⍴⍴X)/BEGIN
       ⎕←'IMPROPER ARGUMENTS'
       →0
      BEGIN:⎕IO←0
       N1←1↑⍴X
       N2←1↓⍴X
       Y←(N2,N1)⍴0
       I1←0
      LOOP1: T1←(-(I1+1))⌽⌽H
       I2←0
      LOOP2: T2←(-(I2+1))⊖⊖T1
       Y[I2;I1]←+/+/X×T2
       →(N2>I2←I2+1)/LOOP2
       →(N1>I1←I1+1)/LOOP1
```

GETDATA

```
     ∇ R←GETDATA NAME;[illegible]
      ⍝
      ⍝ FUNCTION TO READ FREE-FORMATTED DATA (VECTOR, ARRAY, ETC.) FROM CMS DISK
      ⍝ DATA MAY HAVE BEEN CREATED BY APL, PL/I, FORTRAN, EDITOR, ETC.
      ⍝ NUMBERS NEED NOT HAVE DECIMAL POINTS AND ARE DELIMITED BY SPACES
      ⍝ FIRST RECORD IS SHAPE OF THE DATA
      ⍝
       ⎕IO←1
       NAME←NAME,(∧/' '≠NAME)/' DATA'
       REC←NAME,' (370'
       SYSRESP←110 ⎕SVO 'REC'
       SYSRESP←REC
       →(SYSRESP[1]≠0)/ERREXIT
       NREC←SYSRESP[3]-2
       NM←⍎REC
       R←⍳0
       I←0
      LOOP: →(NREC<I←I+1)/CLOSE
       LINE←REC
       LINE[(LINE='-')/⍳⍴LINE]←'¯'
       LINE←[illegible]LINE
       R←R,⍎LINE
       →LOOP
      CLOSE: SYSRESP←⎕SVR 'REC'
       →((×/NM)=⍴R)/EXIT
       'WARNING: INSUFFICIENT OR EXCESS DATA'
      EXIT: R←NM⍴R
       →0
      ERREXIT:'FILE DOES NOT EXIST'
```

PUTDATA

```
     ∇ F PUTDATA NAME;⎕IO;C;SYSRESP;REC;NREC;I
      ⍝
      ⍝ FUNCTION TO WRITE DATA (VECTOR, ARRAY, ETC.) IN FREE FORMAT ON CMS DISK
      ⍝
       ⎕IO←1
       NAME←NAME,(∧/' '≠NAME)/' DATA'
       C←'CMS'
       SYSRESP←100 ⎕SVO 'C'
       C←'ERASE ',NAME
       SYSRESP←⎕SVR 'C'
       REC←NAME,' 370'
       SYSRESP←110 ⎕SVO 'REC'
       REC←' ',⍕⍴F
       NREC←⌈(×/⍴F)÷4
       C←[illegible]4|(×/⍴F),[illegible]
       F←(NREC,4)⍴F
       I←0
      LOOP: →(NREC<I←I+1)/FINISH
       REC←' ',⍕F[I;]
       →LOOP
      FINISH:REC←' ',⍕C↑F[NREC;]
       SYSRESP←⎕SVR 'REC'
```

COVF

```
    ∇ R←SIZE COVF F,⎕IO;K;L;L;K;R1
     ⍝
     ⍝ FUNCTION TO GENERATE TERMS IN 2D COVARIANCE FCN FOR IMAGE
     ⍝
      F←F−MEAN F
      ⎕IO←0
      L←SIZE[0]
      K←SIZE[1]
      R←R1←SIZEρ0
      L←0
    LOOP1:K←0
    LOOP2:R[L;K]←(+/+/((L,K)↓F)×((−(L,K))↓F))÷×/ρF
      R1[L;K]←(+/+/((L,−K)↓F)×((−(L,−K))↓F))÷×/ρF
      →(K>K←K+1)/LOOP2
      →(L>L←L+1)/LOOP1
      R←(⌽ 0 1 ↓R1),R
```

CORR

```
    ∇ RI←SIZE CORR F;⎕IO;R;L;K,L,K
     ⍝
     ⍝ FUNCTION TO ESTIMATE 2D COVARIANCE FUNCTION FOR IMAGE F
     ⍝
      ⎕IO←0
      R←SIZE COVF F
      L←SIZE[0]
      K←SIZE[1]
      RI←(SIZE,K)ρ0
      L←0
    LOOP1:K←0
    LOOP2:RI[L;;K]←(−K)↑(−K)⌽R[L;]
      →(K>K←K+1)/LOOP2
      →(L>L←L+1)/LOOP1
```

MVLEV

```
    ∇ AA←MVLEV RPT,RT;N;NB;NDIM;AA1;AF;AB;AF1,AB1;EF,EB,GAMF,GAMB;DELT,I,⎕IO
     ⍝
     ⍝ RPT IS THE (MATRIX) CORRELATION FUNCTION; PLANE J CONTAINS R(J).
     ⍝ TO OBTAIN FORWARD PARAMETERS:  AF←AA[0;;], BACKWARD  AB←AA[1;;],
     ⍝ ERROR COVARIANCE EF←AF[0;;]. PUT AF[0;;]←I TO OBTAIN INNOVATIONS
     ⍝ FILTER. LIKEWISE FOR BACKWARD PARAMETERS.
     ⍝
      ⎕IO←0
      →((1<(ρRPT)[0])∧3=ρρRPT)/START
      'INVALID ARGUMENT'
      →0
     ⍝
    START:N←¯1+(ρRPT)[0]
      NB←(ρRPT)[1]
      RT← 0 2 1 ⍉RPT
      I←(⍳NB)∘.=⍳NB
      →(N>1)/RECURS
      AF1←(1,NB,NB)ρI
      AB1←AF1
      EF←RPT[0;;]
      EB←EF
      →BGNALG
    RECURS:AA1←MVLEV(N,NB,NB)↑RPT
      AF1←AA1[0;;]
      AB1←AA1[1;;]
      EF←AF1[0;;]
      EB←AB1[0;;]
      [illegible]
      AB1[0;;]←I
    BGNALG:NDIM←N×NB
      DELT←(⍉(NDIM,NB)ρ⊖1 0 0 ↓RPT)+.×(NDIM,NB)ρ⊖AF1
      GAMF←(⌹EB)+.×DELT
      GAMB←(⌹EF)+.×⍉DELT
      AF←(AF1,[0] 0)−(⊖(AB1,[0] 0))+.×GAMF
      AB←(AB1,[0] 0)−(⊖(AF1,[0] 0))+.×GAMB
      NDIM←NB×N+1
      AF[0;;]←(⍉(NDIM,NB)ρRT)+.×(NDIM,NB)ρAF
      AB[0;;]←(⍉(NDIM,NB)ρRPT)+.×(NDIM,NB)ρAB
      AA←AF,[¯0.5] AB
     ∇ A←FF2DLEV R;AA;AF;EN;S;AO;M;⎕IO
      ⍝
      ⍝ FUNCTION TO DO 2D LEVINSON RECURSION - USES FUNCTION MVLEV
      ⍝
       ⎕IO←0
       AA←MVLEV R
       AF←AA[0;;]
       EN←AF[0;;]
       M←(⍴R)[1]
       AF[0;;]←(⍳M)∘.=⍳M
       S←M↑ 1 0
       AO←(⌹EN)+.×S
       A←(AF+.×AO)-AO[0]
       A[0;0]←-AO[0]
       ∇
       ∇COVF[⎕]∇
       ∇ R←SIZE COVF F;⎕IO;K̲;L̲;L;K;R1
      ⍝
      ⍝ FUNCTION TO GENERATE TERMS IN 2D COVARIANCE FCN FOR IMAGE
      ⍝
       F←F-MEAN F
       ⎕IO←0
       L̲←SIZE[0]
       K̲←SIZE[1]
       R←R1←SIZE⍴0
       L←0
      LOOP1: K←0
      LOOP2: R[L;K]←(+/+/((L,K)↓F)×((-(L,K))↓F))-×/⍴F
       R1[L;K]←(+/+/((L,-K)↓F)×((-(L,-K))↓F))-×/⍴F
       →(K̲>K←K+1)/LOOP2
       →(L̲>L←L+1)/LOOP1
       R←(⊖ 0 1 ↓R1),R
```

APPENDIX B

DERIVATION OF THE POWER SPECTRUM AND AUTOCORRELATION FUNCTION FOR THE TWO POLE AUTOREGRESSIVE MODEL

Power Spectrum

$$|H(e^{j\omega})|^2 = H(e^{j\omega})\cdot(H(e^{j\omega}))^* = H(e^{j\omega})H(e^{-j\omega})$$

In this case:

$$H(e^{j\omega}) = \frac{1}{1+\alpha e^{-j\omega}} \qquad H(e^{-j\omega}) = \frac{1}{1+\alpha e^{j\omega}}$$

Calculating $H(e^{j\omega})H(e^{-j\omega})$:

$$\frac{1}{(1+\alpha e^{-j\omega})(1+\alpha e^{j\omega})} = \frac{1}{1+\alpha e^{-j\omega}+\alpha e^{j\omega}+\alpha^2} = \frac{1}{1+2\alpha\cos(\omega)+\alpha^2}$$

The final result is:

$$S_y(\omega) = |H(e^{j\omega})|^2 = \frac{1}{(1+\alpha^2)+2\alpha\cos(\omega)} \qquad \text{(B.1)}$$

Autocorrelation Function

Starting with H(z) for this case:

$$H(z) = \frac{1}{1+\alpha z^{-1}}$$

Per Ref. 7:p. 158:

$Z^{-1}[H(z)] = h(n) = (-\alpha)^n\cdot u(n)$ for $\alpha < 1$
(u(n) is the unit step function)

Per Ref. 5:pp. 391-395, for the white noise input case:

$$R_y(\ell) = \sum_{n=-\infty}^{\infty} h(n)\cdot h(n-\ell) \qquad \text{(B.2)}$$

Substituting h(n) above into Eq. B.2

$$R_y(\ell) = \sum_{n=\ell}^{\infty} (-\alpha)^n \cdot (-\alpha)^{n-\ell} \qquad \ell \geq 0$$

$$= \frac{1}{(-\alpha)^{\ell}} \sum_{n=\ell}^{\infty} (-\alpha)^{2n} \qquad \ell \geq 0$$

The summation term may also be expressed as:

$$\sum_{n=\ell}^{\infty} (-\alpha)^{2n} = \sum_{n=0}^{\infty} (-\alpha)^{2n} - \sum_{n=0}^{\ell-1} (-\alpha)^{2n}$$

Per Ref. 12:p. 8, the summation terms on the right are equal to:

$$\sum_{n=0}^{\infty} (-\alpha)^{2n} = \frac{1}{1-(-\alpha)^2} \quad (\alpha < 1) \qquad \sum_{n=0}^{\ell-1} (-\alpha)^{2n} = \frac{1-(-\alpha)^{2\ell}}{1-(-\alpha)^2} \quad (\alpha < 1)$$

As a result:

$$\sum_{n=\ell}^{\infty} (-\alpha)^{2n} = \frac{1}{1-(-\alpha^2)} - \frac{1-(-\alpha)^{2\ell}}{1-(-\alpha)^2} = \frac{(-\alpha)^{2\ell}}{1-(-\alpha)^2} \quad (\alpha < 1)$$

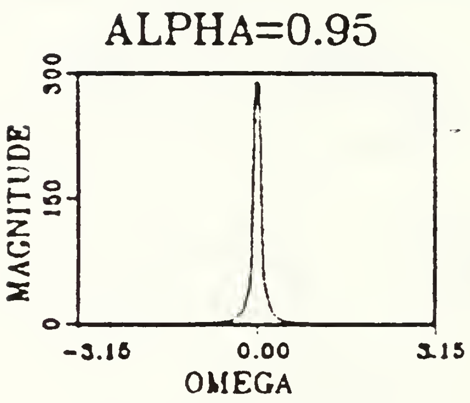
Substituting and using $(-\alpha)^2 = \alpha^2$ yields:

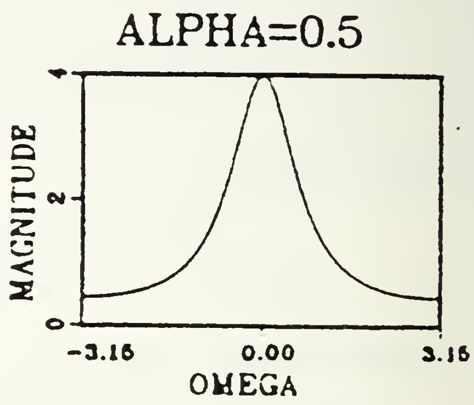
$$R_y(\ell) = \frac{1}{(-\alpha)^{\ell}} \cdot \frac{(-\alpha)^{2\ell}}{1-\alpha^2} = \frac{(-\alpha)^{\ell}}{1-\alpha^2} \quad \ell \geq 0 \text{ and } \alpha < 1 \qquad (B.3)$$

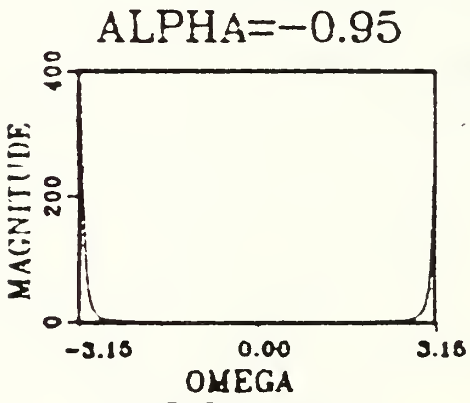
APPENDIX C

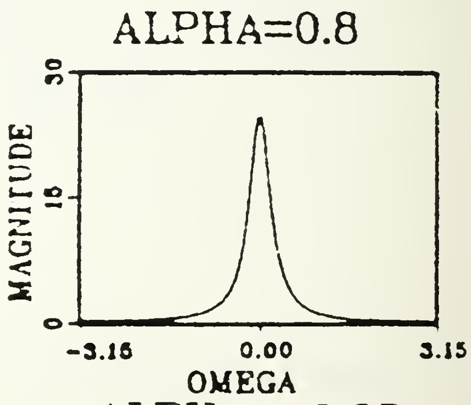
<u>GRAPHICAL RESULTS FOR THE POWER SPECTRUM AND AUTOCORRELATION FUNCTION FOR THE TWO POLE AUTOREGRESSIVE MODEL</u>

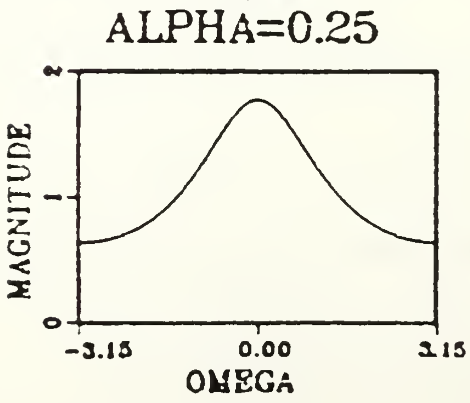
Power Spectrum

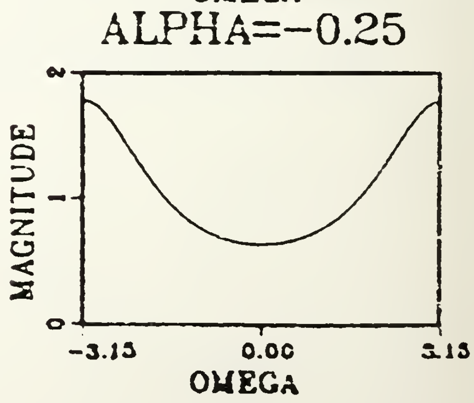


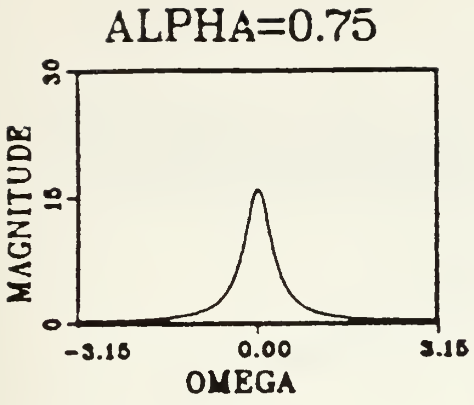


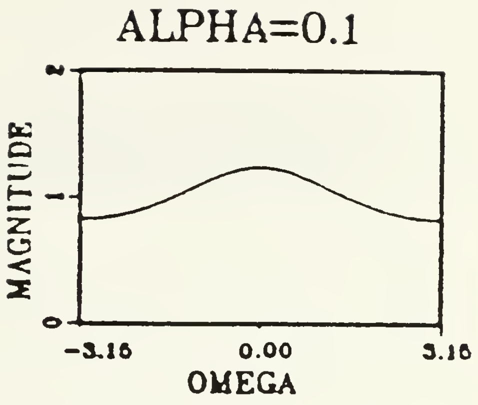




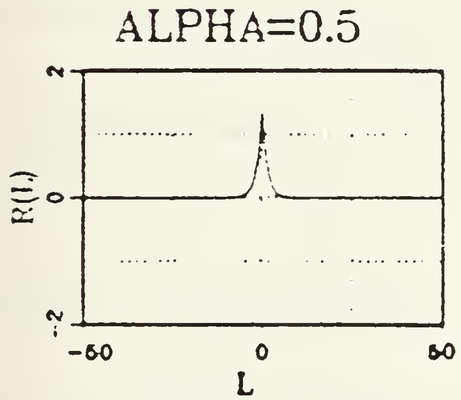


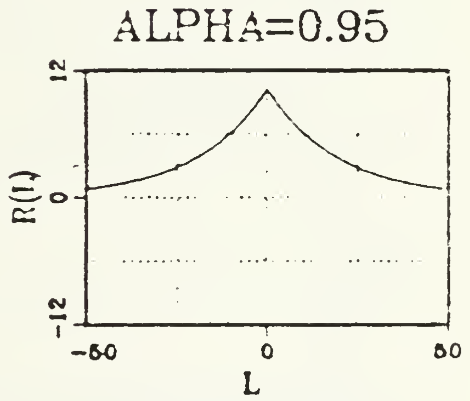


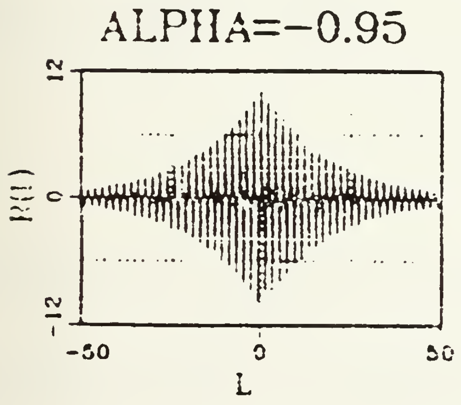


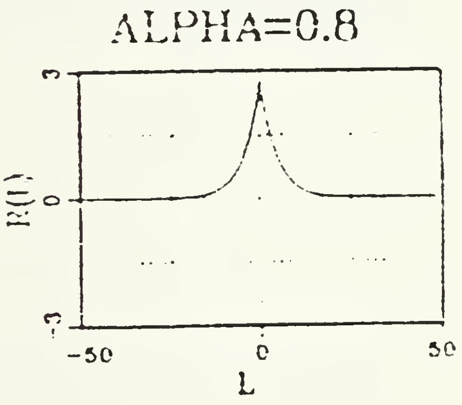


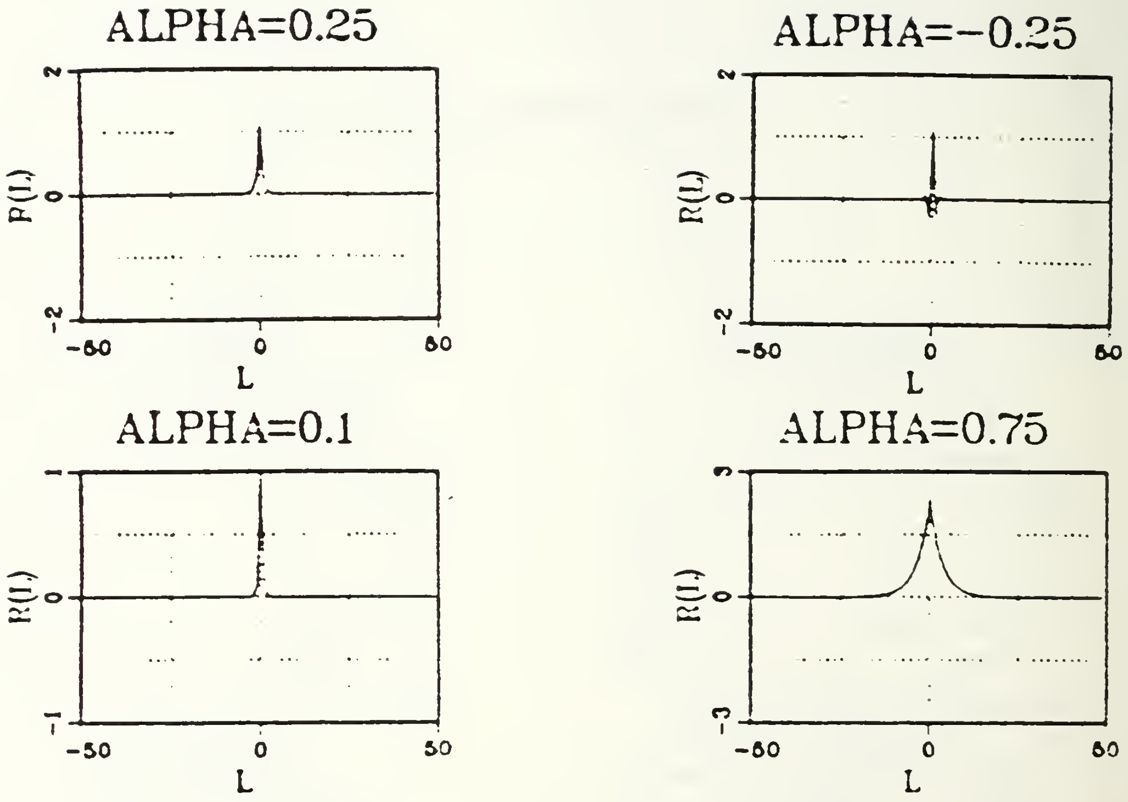
Autocorrelation Function










ALPHA=0.25
R(L)
2
0
-2
-50
0
50
L
ALPHA=-0.25
R(L)
2
0
-2
-50
0
50
L
ALPHA=0.1
R(L)
1
0
-1
-50
0
50
L
ALPHA=0.75
R(L)
3
0
-3
-50
0
50
L

APPENDIX D

DERIVATION OF THE POWER SPECTRUM AND AUTOCORRELATION FUNCTION FOR THE FOUR POLE AUTOREGRESSIVE MODEL

Power Spectrum

$$|H(e^{j\omega})|^2 = H(e^{j\omega})\cdot H(e^{-j\omega})$$

In this case:

$$H(e^{-j\omega}) = \frac{1}{(1-\alpha e^{-j\theta}e^{j\omega})(1-\alpha e^{j\theta}e^{j\omega})} = \frac{1}{1-\alpha e^{j(\theta+\omega)}-\alpha e^{-j(\theta-\omega)}+\alpha^2 e^{j2\omega}}$$

$$H(e^{j\omega}) = \frac{1}{(1-\alpha e^{-j\theta}e^{-j\omega})(1-\alpha e^{j\theta}e^{-j\omega})} = \frac{1}{1-\alpha e^{-j(\theta+\omega)}-\alpha e^{j(\theta-\omega)}+\alpha^2 e^{-j2\omega}}$$

Multiplying the above expressions yields:

$$H(e^{j\omega})H(e^{-j\omega}) = \frac{1}{\begin{aligned}&1-\alpha e^{j(\theta-\omega)}-\alpha e^{-j(\theta+\omega)}+\alpha^2 e^{-j2\omega}-\alpha e^{j(\theta+\omega)}+\alpha^2 e^{j2\theta}+\alpha^2\\ &-\alpha^3 e^{j(\theta-\omega)}-\alpha e^{-j(\theta-\omega)}+\alpha^2+\alpha^2 e^{-j2\theta}-\alpha^3 e^{-j(\theta-\omega)}\\ &+\alpha^2 e^{j2\omega}-\alpha^3 e^{j(\theta+\omega)}-\alpha^3 e^{-j(\theta+\omega)}+\alpha^4\end{aligned}}$$

Combining terms:

$$H(e^{j\omega})\cdot H(e^{-j\omega}) = \frac{1}{\begin{aligned}&1+2\alpha^2+\alpha^4-\alpha^3(e^{-j(\theta-\omega)}+e^{j(\theta-\omega)})+\alpha^2(e^{j2\theta}+e^{-j2\theta})\\ &-\alpha^3(e^{j(\theta+\omega)}+e^{-j(\theta+\omega)})+\alpha^2(e^{j2\omega}+e^{-j2\omega})\\ &-\alpha(e^{j(\theta-\omega)}+e^{-j(\theta-\omega)})-\alpha(e^{j(\theta+\omega)}+e^{-j(\theta+\omega)})\end{aligned}}$$

Using Euler's relation and combining terms:

$$H(e^{j\omega})\cdot H(e^{-j\omega}) = \frac{1}{\begin{aligned}&1+\alpha^4+2(\alpha^2-\cos(\theta-\omega)[\alpha^3+\alpha]+\alpha^2\cos(2\theta)-\cos(\theta+\omega)\\ &\times[\alpha^3+\alpha]+\alpha^2\cos(2\omega))\end{aligned}}$$

Using $\cos(\theta-\omega) + \cos(\theta+\omega) = 2\cos(\theta)\cos(\omega)$, and since $S_y(\omega) = H(e^{j\omega})\cdot H(e^{-j\omega})\cdot\sigma^2$, with $\sigma^2 = 1$ the final result is:

$$S_y(\omega) = \frac{1}{1+\alpha^4+2(\alpha^2-2[\alpha^3-\alpha]\cos(\theta)\cos(\omega)+\alpha^2(\cos(2\theta)+\cos(2\omega)))} \qquad \text{(D.1)}$$

Autocorrelation Function

$$H(z) = \frac{1}{1-\alpha e^{j\theta}z^{-1}-\alpha e^{-j\theta}z^{-1}+\alpha^2 z^{-2}} = \frac{1}{1-(e^{j\theta}+e^{-j\theta})z^{-1}+\alpha^2 z^{-2}}$$

Using Euler's relation:

$$H(z) = \frac{1}{1-2\alpha\cos(\theta)z^{-1}+\alpha^2 z^{-2}} \qquad |z| > |\alpha|$$

Per Ref. 7:pp. 204-216, partial fraction expansion can be used to find the inverse Z transform. To do so H(z) can be expressed in the form:

$$H(z) = \frac{1}{(1-\alpha e^{-j\theta}z^{-1})(1-\alpha e^{j\theta}z^{-1})} = \frac{z}{(z-\alpha e^{-j\theta})(z-\alpha e^{j\theta})}$$

Using the partial fraction expansion and table look up [Ref. 7:p. 158] yields:

$$h(n) = \frac{\alpha^n}{\sin(\theta)}\cos(n\theta+\theta-\tfrac{\pi}{2})\cdot u(n) \quad \text{for} \quad \alpha < 1 \qquad \text{(D.2)}$$

Since $\cos(\theta - \frac{\pi}{2}) = \sin(\theta)$, the final expression for h(n) is:

$$h(n) = \frac{\alpha^n}{\sin(\theta)}\sin((n+1)\theta)\cdot u(n) \qquad \alpha < 1 \qquad \text{(D.3)}$$

For simplicity in further derivation of $R_y(\ell)$ based on $h(n)$, Eq. D.2 will be used.

Using the expression for the autocorrelation function of a random process represented by the above filter with a white noise input [Ref. 4:pp. 391-395]:

$$R_y(\ell) = \sum_{n=-\infty}^{\infty} h(n)\cdot h(n-\ell) = \frac{\alpha^{-\ell}}{\sin^2(\theta)} \sum_{n=\ell}^{\infty} \alpha^{2n}\cdot\cos(n\theta+\theta-\tfrac{\pi}{2})$$
$$\cdot\cos((n-\ell)\theta+\theta-\tfrac{\pi}{2}) \qquad \text{(D.4)}$$

$n = \ell$ in the summation index since $h(n)$ is causal. ℓ is assumed to be greater than zero here. For $\ell < 0$, $R_y(\ell) = R_y(-\ell)$ by symmetry of the autocorrelation function [Ref. 5:p. 388], so we can proceed assuming only positive values of ℓ.

Using the trigonometric identity for a product of cosines:

$$R_y(\ell) = \frac{\alpha^{-\ell}}{\sin^2(\theta)} \sum_{n=\ell}^{\infty} \alpha^{2n}[\tfrac{1}{2}\cdot\cos(\ell\theta)+\tfrac{1}{2}\cos(2n\theta-\ell\theta+2\theta-\pi)] \quad \ell \geq 0$$

$$R_y(\ell) = \frac{\cos(\ell\theta)\cdot\alpha^{-\ell}}{2\sin^2(\theta)}\cdot\sum_{n=\ell}^{\infty}\alpha^{2n} + \frac{\alpha^{-\ell}}{2\sin^2(\theta)}\sum_{n=\ell}^{\infty}\alpha^{2n}\cdot(2n\theta-\ell\theta+2\theta-\pi) \qquad \text{(D.5)}$$
$$\ell \geq 0$$

Using $\sum_{n=\ell}^{\infty} \rightarrow \sum_{n=0}^{\infty} - \sum_{n=0}^{\ell-1}$ and standard geometric progression identities [Ref.]2:p. 8]:

$$\sum_{n=0}^{\infty} \alpha^n = \frac{1}{1-\alpha} \quad \text{and} \quad \sum_{n=0}^{\ell-1} \alpha^n = \frac{1-\alpha^{\ell}}{1-\alpha}$$

For the first term in $R_y(\ell)$

$$\sum_{n=\ell}^{\infty} \alpha^{2n} = \sum_{n=0}^{\infty} \alpha^{2n} - \sum_{n=0}^{\ell-1} \alpha^{2n} = \frac{1}{1-\alpha^2} - \frac{1-\alpha^{2\ell}}{1-\alpha^2}$$

$$= \frac{\alpha^{2\ell}}{1-\alpha^2} \qquad \text{(D.6)}$$

For the second term in $R_y(\ell)$:

let $\phi = -\ell\theta + 2\theta - \pi$

Using Euler's relation:

$$\cos(2n\theta+\phi) = \frac{e^{j(2n\theta+\phi)} + e^{-j(2n\theta+\phi)}}{2}$$

$$\sum_{n=\ell}^{\infty} \alpha^{2n}\cdot\cos(2n\theta+\phi) = \frac{1}{2}\sum_{n=\ell}^{\infty} \alpha^{2n}[e^{j(2n\theta+\phi)} + e^{-j(2n\theta+\phi)}]$$

$$= \frac{1}{2}\sum_{n=\ell}^{\infty} \alpha^{2n}\cdot e^{j(2n\theta+\phi)} + \frac{1}{2}\sum_{n=\ell}^{\infty} \alpha^{2n}\cdot e^{-j(2n\theta+\phi)} \qquad \text{(D.7)}$$

For large n, it is evident that the α^{2n} term will tend to make the term in each sum approach 0 for $\alpha < 1$, and thus ensures convergence and a closed form expression for each term. Pursuing the mathematics required to find this closed form expression we have:

$$\frac{1}{2}\sum_{n=\ell}^{\infty} \alpha^{2n}\cdot e^{j(2n\theta+\phi)} = \frac{e^{j\phi}}{2}\sum_{n=\ell}^{\infty} (\alpha e^{j\theta})^{2n} = \frac{e^{j\phi}}{2}[\sum_{n=0}^{\infty} (\alpha e^{j\theta})^{2n} - \sum_{n=0}^{\ell-1} (\alpha e^{j\theta})^{2n}]$$

$$\frac{1}{2}\sum_{n=\ell}^{\infty}\alpha^{2n}\cdot e^{j(2n\theta+\phi)} = \frac{e^{j\phi}}{2}\left[\frac{1}{1-(\alpha e^{j\theta})^2} - \frac{1-(\alpha e^{j\theta})^{2\ell}}{1-(\alpha e^{j\theta})^2}\right]$$

$$= \frac{\alpha^{2\ell}\cdot e^{j(2\theta\ell+\phi)}}{2(1-(\alpha e^{j\theta})^2)} \qquad \text{(D.8)}$$

For the conjugate term we must have:

$$\frac{1}{2}\sum_{n=\ell}^{\infty}\alpha^{2n}\cdot e^{-j(2n\theta+\phi)} = \frac{\alpha^{2\ell}\cdot e^{-j(2\theta\ell+\phi)}}{2(1-(\alpha e^{-j\theta})^2)} \qquad \text{(D.9)}$$

Next a common denominator must be found to sum these two terms:

$$\frac{\alpha^{2\ell}\cdot e^{j(2\theta\ell+\phi)}}{2(1-(\alpha e^{j\theta})^2)} \cdot \frac{(1-(\alpha e^{-j\theta})^2)}{(1-(\alpha e^{-j\theta})^2)}$$

$$= \frac{\alpha^{2\ell}e^{j(2\theta\ell+\phi)}-\alpha^{(2\ell+2)}e^{j(2\theta\ell-2\theta+\phi)}}{2(1-(\alpha e^{j\theta})^2-(\alpha e^{-j\theta})^2+\alpha^4)}$$

$$\frac{\alpha^{2\ell}\cdot e^{-j(2\theta\ell+\phi)}}{2(1-(\alpha e^{-j\theta})^2)} \cdot \frac{(1-(\alpha e^{j\theta})^2)}{(1-(\alpha e^{j\theta})^2)}$$

$$= \frac{\alpha^{2\ell}e^{-j(2\theta\ell+\phi)}-\alpha^{(2\ell+2)}e^{-j(2\theta\ell-2\theta+\phi)}}{2(1-(\alpha e^{j\theta})^2-(\alpha e^{-j\theta})^2+\alpha^4)}$$

By Euler's relation:

$$1-(\alpha e^{j\theta})^2-(\alpha e^{-j\theta})^2+\alpha^4 = 1-2\alpha^2\cos(2\theta)+\alpha^4$$

Adding the terms with the common denominator yields:

$$\frac{\alpha^{2\ell}[e^{j(2\theta\ell+\phi)}-\alpha^2 e^{j(2\theta\ell-2\theta+\phi)}+e^{-j(2\theta\ell+\phi)}-\alpha^2 e^{-j(2\theta\ell-2\theta+\phi)}]}{2(1-2\alpha^2\cos(2\theta)+\alpha^4)}$$

Again using Euler's relation the above expression reduces to:

$$\frac{\alpha^{2\ell}[\cos(2\theta\ell+\phi)-\alpha^2\cos(2\theta\ell-2\theta+\phi)]}{1-2\alpha^2\cos(2\theta)+\alpha^4} \tag{D.10}$$

which is the sum of the last two terms in Eq. (D.7). Substituting Eq. (D.10) and Eq. (D.6) into Eq. (D.5) yields:

$$R_Y(\ell) = \frac{\cos(\ell\theta)\cdot\alpha^{-\ell}}{2\sin^2(\theta)}\left[\frac{\alpha^{2\ell}}{1-\alpha^2}\right] + \frac{\alpha^{-\ell}}{2\sin^2(\theta)}\left[\frac{\alpha^{2\ell}[\cos(2\theta\ell+\phi)-\alpha^2\cos(2\theta\ell-2\theta+\phi)]}{1-2\alpha^2\cos(2\theta)+\alpha^4}\right] \tag{D.11}$$

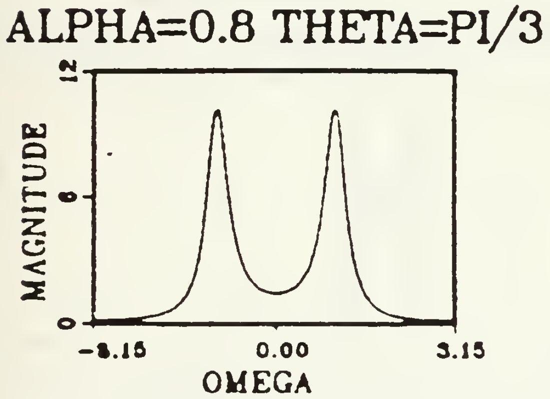
Combining and canceling terms and substituting for ϕ and noticing that the same result must hold for $\ell < 0$ we have:

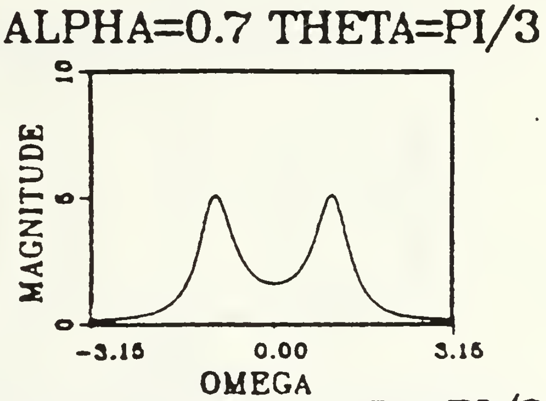
$$R_Y(\ell) = \frac{\alpha^{\ell}}{2\sin^2(\theta)}\left[\frac{\cos(|\ell|\theta)}{1-\alpha^2} + \frac{\cos((2+|\ell|)\theta-\pi)-\alpha^2\cos(|\ell|\theta-\pi)}{1-2\alpha^2\cos(2\theta)+\alpha^4}\right] \tag{D.12}$$

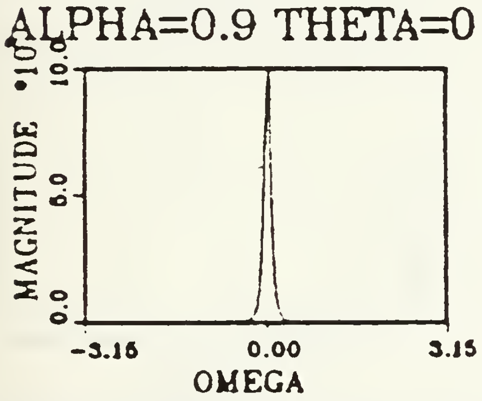
APPENDIX E

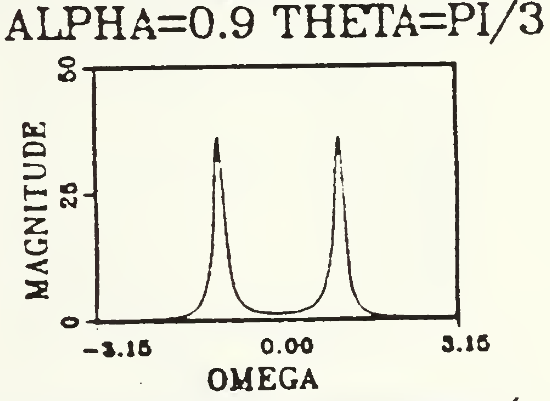
GRAPHICAL RESULTS FOR THE POWER SPECTRUM AND AUTOCORRELATION FUNCTION FOR THE FOUR POLE AUTOREGRESSIVE MODEL

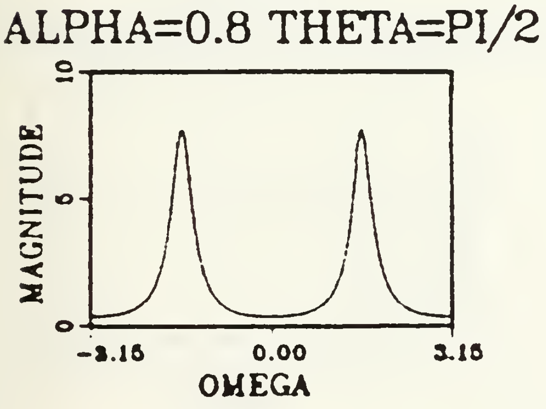
Power Spectrum

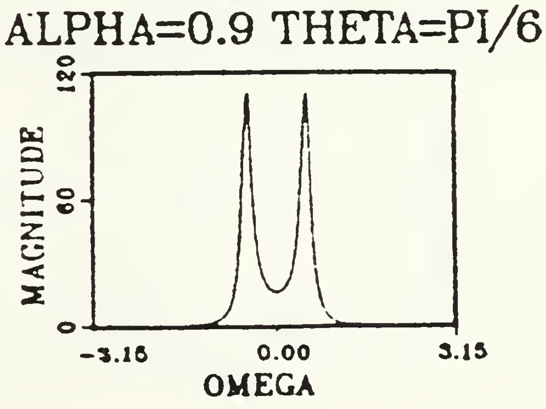




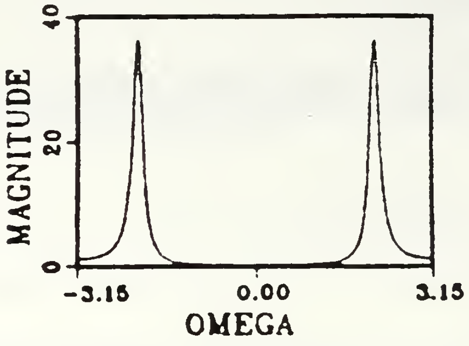




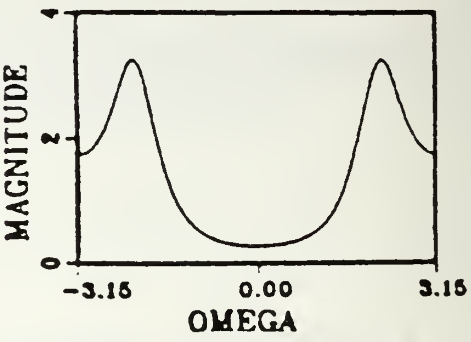




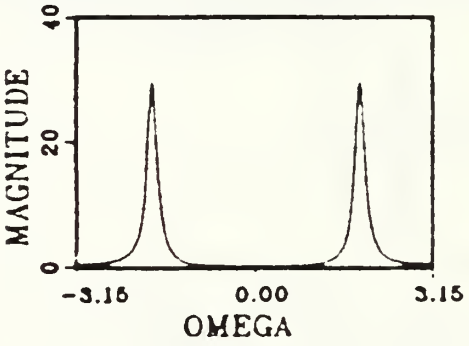
ALPHA=0.9 THETA=2*PI/3



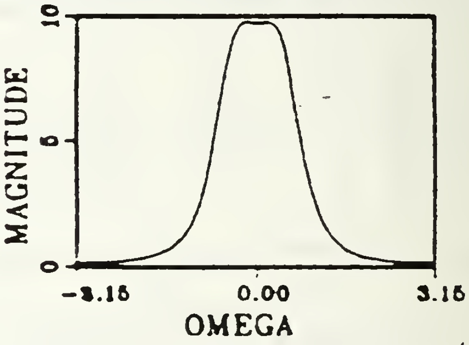
ALPHA=0.6 THETA=2*PI/3



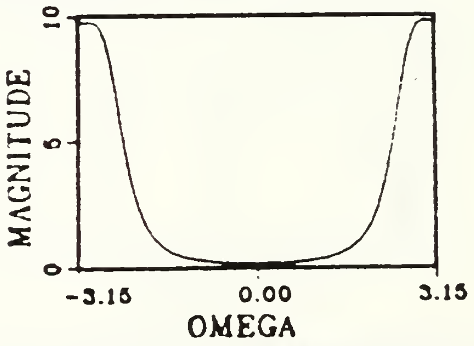
ALPHA=0.9 THETA=7*PI/12



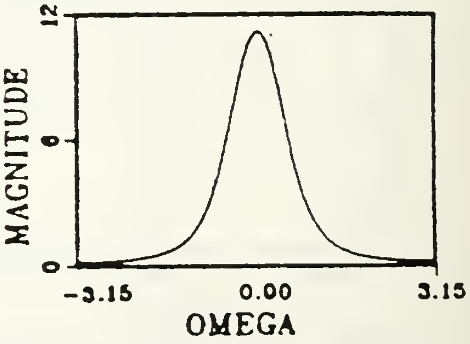
ALPHA=0.6 THETA=PI/6



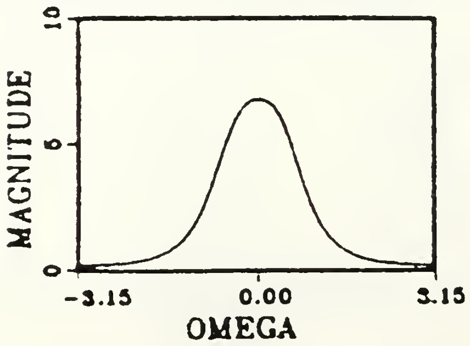
ALPHA=0.6 THETA=5*PI/6



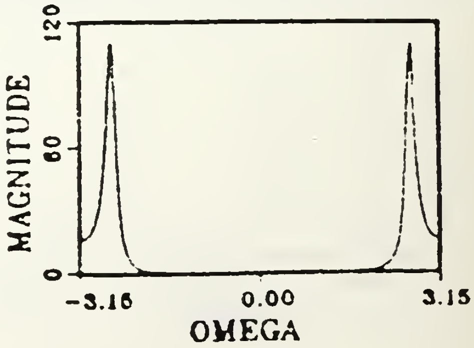
ALPHA=0.5 THETA=PI/10

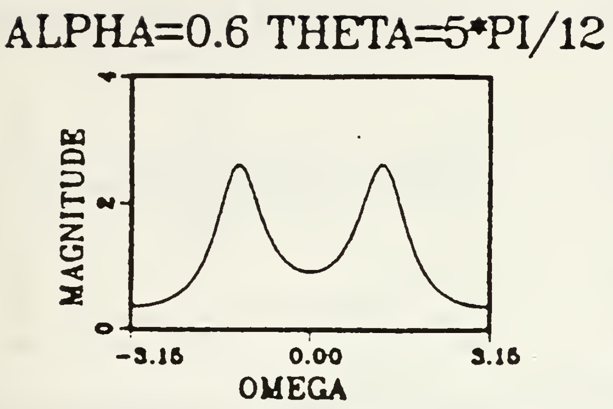


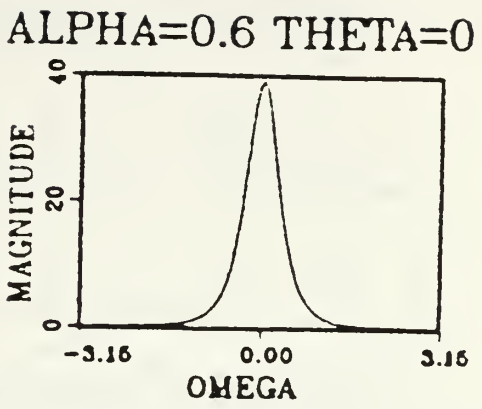
ALPHA=0.5 THETA=PI/6

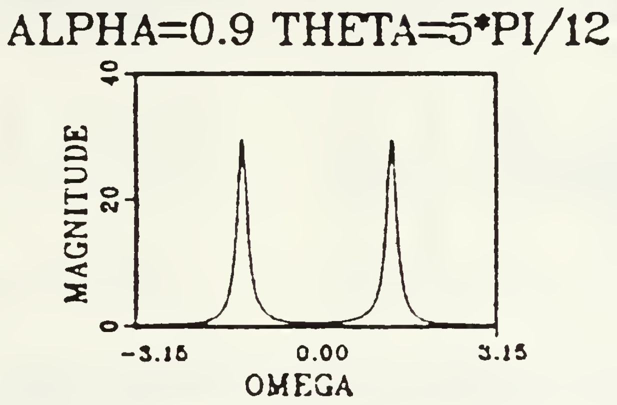


ALPHA=0.9 THETA=5*PI/6

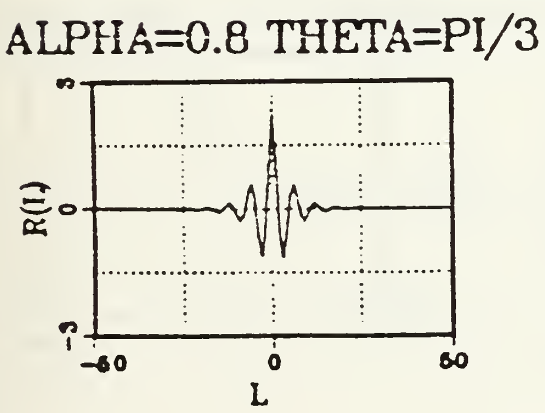


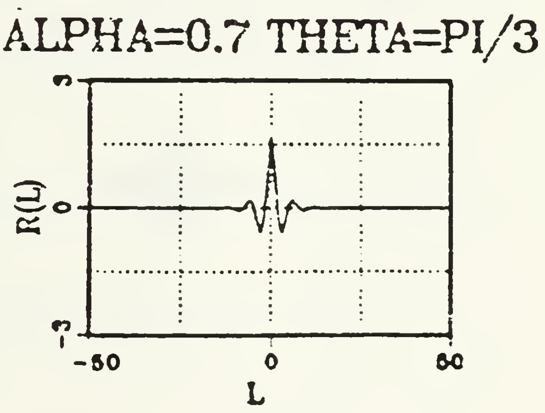


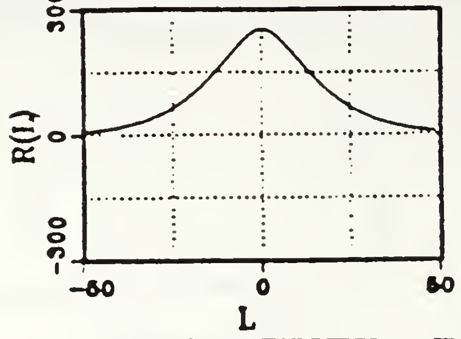


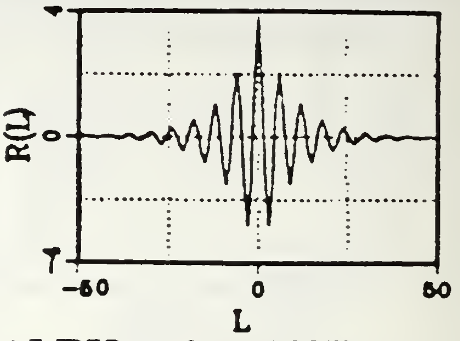


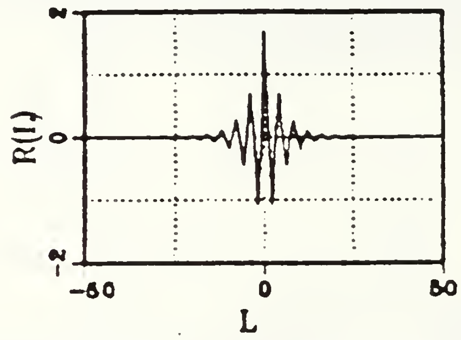
Autocorrelation Function

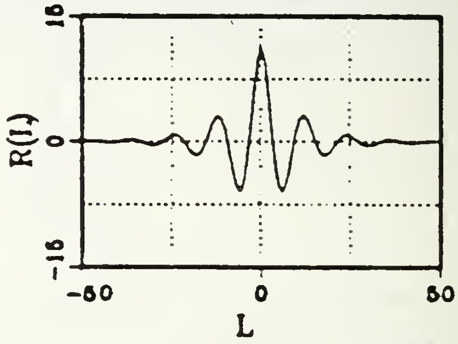


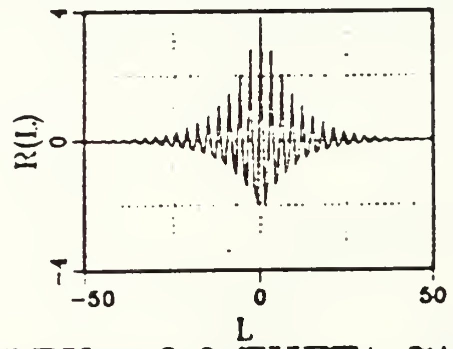


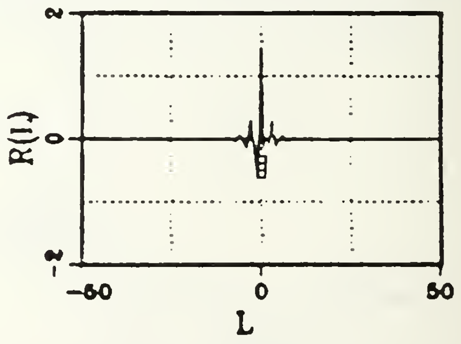

ALPHA=0.9 THETA=0

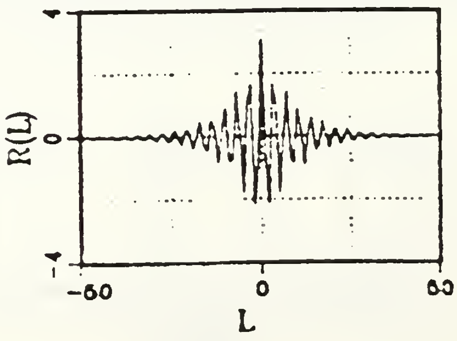

ALPHA=0.9 THETA=PI/3

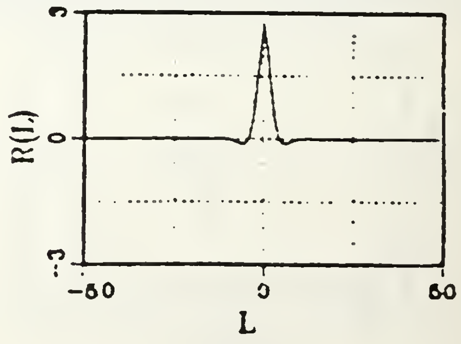

ALPHA=0.8 THETA=PI/2

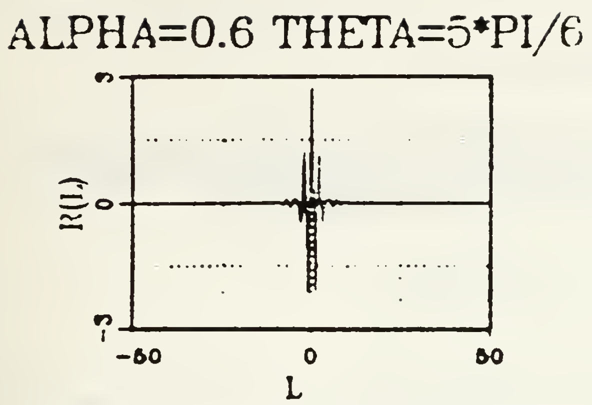

ALPHA=0.9 THETA=PI/6

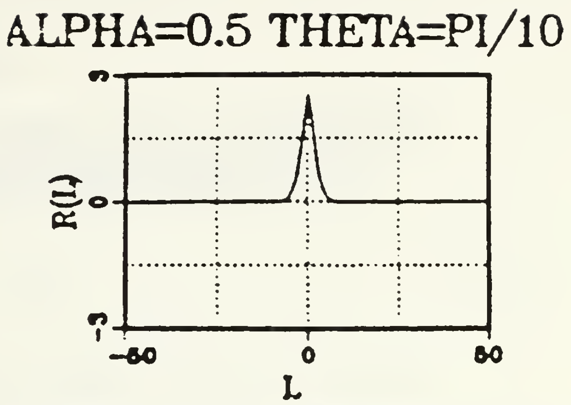

ALPHA=0.9 THETA=2*PI/3

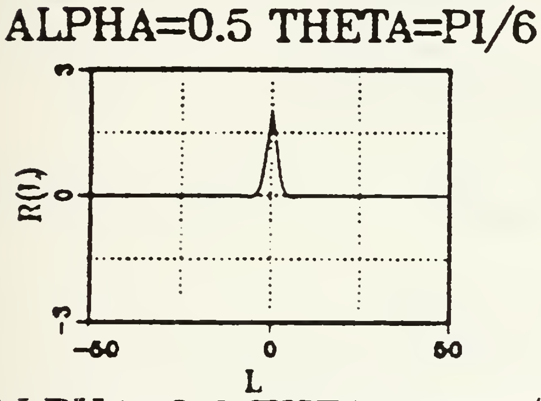

ALPHA=0.6 THETA=2*PI/3

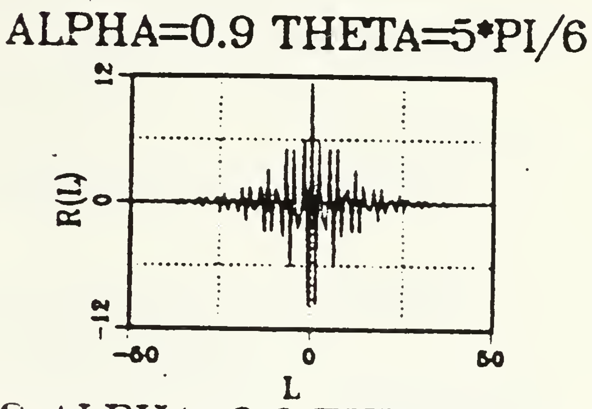

ALPHA=0.9 THETA=7*PI/12

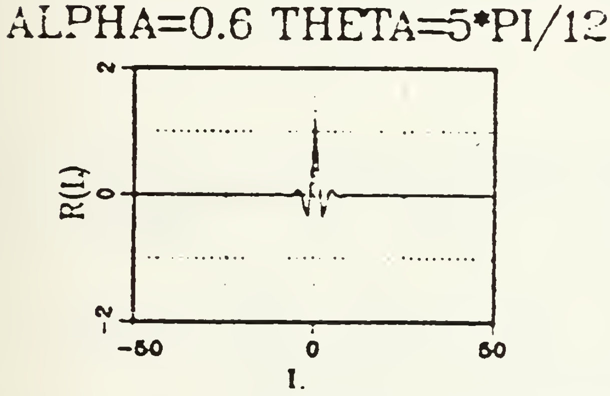

ALPHA=0.6 THETA=PI/6

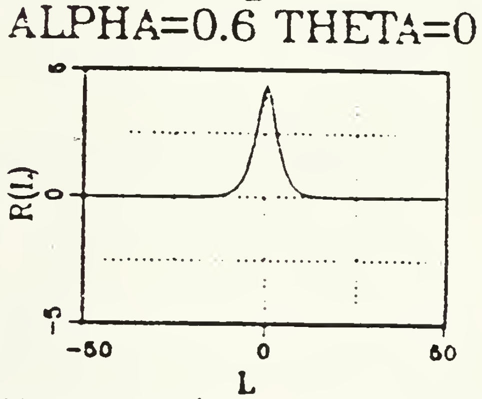

ALPHA=0.6 THETA=5*PI/6
R(L)
3
0
-3
-50
0
50
L

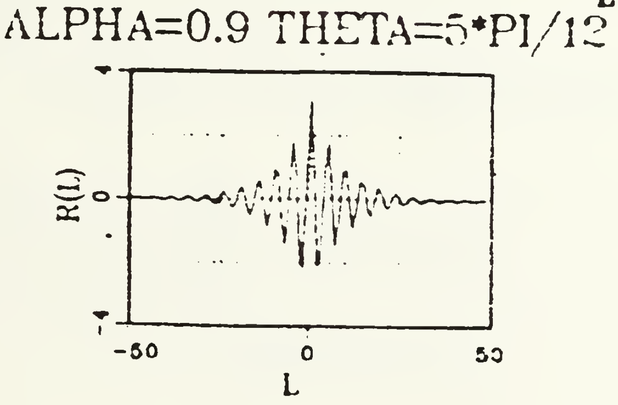

ALPHA=0.5 THETA=PI/10
R(L)
3
0
-3
-50
0
50
L


ALPHA=0.5 THETA=PI/6
R(L)
3
0
-3
-50
0
50
L


ALPHA=0.9 THETA=5*PI/6
R(L)
12
0
-12
-50
0
50
L


ALPHA=0.6 THETA=5*PI/12
R(L)
2
0
-2
-50
0
50
L


ALPHA=0.6 THETA=0
R(L)
5
0
-5
-50
0
50
L


ALPHA=0.9 THETA=5*PI/12
R(L)
4
0
-4
-50
0
50
L

APPENDIX F

DERIVATION OF THE POWER SPECTRUM AND AUTOCORRELATION FUNCTION FOR THE FOUR POLE AUTOREGRESSIVE MODEL (WITH TWO POLES ON THE REAL AXIS)

Power Spectrum

Using initial results from Appendix D with the necessary modifications (including $\theta = 0$) we have:

$$H(e^{-j\omega}) = \frac{1}{1-\alpha_a e^{j\omega}-\alpha_b e^{j\omega}+\alpha_a\alpha_b e^{j2\omega}}$$

$$H(e^{j\omega}) = \frac{1}{1-\alpha_a e^{-j\omega}-\alpha_b e^{-j\omega}+\alpha_a\alpha_b e^{-j2\omega}}$$

Therefore:

$$\begin{aligned} H(e^{j\omega})\cdot H(e^{-j\omega}) = \frac{1}{\begin{aligned}&1-\alpha_a e^{-j\omega}-\alpha_b d^{-j\omega}+\alpha_a\alpha_b e^{-j2\omega}-\alpha_a e^{j\omega}+\alpha_a^2+\alpha_a\alpha_b\\ &-\alpha_a^2\alpha_b e^{-j\omega}-\alpha_b e^{j\omega}+\alpha_a\alpha_b+\alpha_b^2-\alpha_a\alpha_b^2 e^{-j\omega}\\ &+\alpha_a\alpha_b e^{j2\omega}-\alpha_a^2\alpha_b e^{j\omega}-\alpha_a\alpha_b^2 e^{j\omega}+\alpha_a^2\alpha_b^2\end{aligned}}\end{aligned}$$

Combining terms and using Euler's relation:

$$H(e^{j\omega})\cdot H(e^{-j\omega}) = \frac{1}{\begin{aligned}&1-(\alpha_a+\alpha_b+\alpha_a^2\alpha_b+\alpha_a\alpha_b^2)\cdot 2\cos(\omega)+2\alpha_a\alpha_b\cos(2\omega)+\alpha_a^2\\ &+2\alpha_a\alpha_b+\alpha_b^2+\alpha_a^2\alpha_b^2\end{aligned}}$$

Assuming $\sigma^2 = 1$ and since $S_y(\omega) = H(e^{j\omega})\cdot H(e^{-j\omega})\cdot\sigma^2$, the final result is:

$$S_y(\omega) = \frac{1}{1-2(\alpha_a+\alpha_b+\alpha_a^2\alpha_b+\alpha_a\alpha_b^2)\cos(\omega)+2\alpha_a\alpha_b\cos(2\omega)+\alpha_a^2+2\alpha_a\alpha_b + \alpha_b^2+\alpha_a^2\alpha_b^2} \qquad (F.1)$$

Autocorrelation Function

$$H(z) = \frac{1}{1-\alpha_a e^{j\theta}z^{-1}-\alpha_b e^{-j\theta}z^{-1}+\alpha_a\alpha_b z^{-2}}$$

Letting $\theta = 0$

$$H(z) = \frac{1}{1-(\alpha_a+\alpha_b)z^{-1}+\alpha_a\alpha_b z^{-2}} = \frac{z^2}{z^2-(\alpha_a+\alpha_b)z+\alpha_a\alpha_b}$$

$$|z| > |\alpha|$$

Expanding in terms of partial fractions we have:

$$H(z) = \frac{\alpha_a}{\alpha_a-\alpha_b}\cdot\frac{z}{z-\alpha_a} + \frac{\alpha_b}{\alpha_b-\alpha_a}\cdot\frac{z}{z-\alpha_b}$$

This corresponds to the impulse response

$$h(n) = \left(\frac{\alpha_a}{\alpha_a-\alpha_b}\cdot\alpha_a^n + \frac{\alpha_b}{\alpha_b-\alpha_a}\cdot\alpha_b^n\right)\cdot u(n) \qquad \alpha < 1 \qquad (F.2)$$

Proceeding as in Appendix D:

$$R_y(\ell) = \sum_{n=-\infty}^{\infty} h(n)\cdot h(n-\ell) = \sum_{n=\ell}^{\infty} \left(\frac{\alpha_a^{n+1}}{\alpha_a-\alpha_b} + \frac{\alpha_b^{n+1}}{\alpha_b-\alpha_a}\right)\left(\frac{\alpha_a^{n+1-\ell}}{\alpha_a-\alpha_b} + \frac{\alpha_b^{n+1-\ell}}{\alpha_b-\alpha_a}\right)$$

$$(\ell \geq 0)$$

$$R_y(\ell) = \sum_{n=\ell}^{\infty} \left[\frac{\alpha_a^{2n+2-\ell}}{(\alpha_a-\alpha_b)^2} + \frac{\alpha_a^{n+1}\cdot\alpha_b^{n+1-\ell}}{(\alpha_a-\alpha_b)(\alpha_b-\alpha_a)} + \frac{\alpha_a^{n+1-\ell}\cdot\alpha_b^{n+1}}{(\alpha_b-\alpha_a)(\alpha_a-\alpha_b)} + \frac{\alpha_b^{2n+2-\ell}}{(\alpha_b-\alpha_a)^2}\right] \quad (\ell \geq 0) \qquad \text{(F.3)}$$

Since $(\alpha_a-\alpha_b) = -(\alpha_b-\alpha_a)$:

$$(\alpha_a-\alpha_b)^2 = (\alpha_b-\alpha_a)^2 = -(\alpha_a-\alpha_b)(\alpha_b-\alpha_a) = -(\alpha_b-\alpha_a)(\alpha_a-\alpha_b)$$

So:

$$R_y(\ell) = \frac{1}{(\alpha_a-\alpha_b)^2}\left[\alpha_a^{2-\ell}\sum_{n=\ell}^{\infty}\alpha_a^{2n} - \alpha_a\alpha_b^{1-\ell}\sum_{n=\ell}^{\infty}(\alpha_a\alpha_b)^n - \alpha_b\alpha_a^{1-\ell}\sum_{n=\ell}^{\infty}(\alpha_b\alpha_a)^n + \alpha_b^{2-\ell}\sum_{n=\ell}^{\infty}\alpha_b^{2n}\right] \quad (\ell \geq 0)$$

Continuing with the same principles and assumptions as in Appendix D, we have:

$$\sum_{n=\ell}^{\infty}\alpha^{2n} = \sum_{n=0}^{\infty}\alpha^{2n} - \sum_{n=0}^{\ell-1}\alpha^{2n} = \frac{1}{1-\alpha^2} - \frac{1-\alpha^{2\ell}}{1-\alpha^2} = \frac{\alpha^{2\ell}}{1-\alpha^2}$$

$$\sum_{n=\ell}^{\infty}\alpha^{n} = \sum_{n=0}^{\infty}\alpha^{n} - \sum_{n=0}^{\ell-1}\alpha^{n} = \frac{1}{1-\alpha} - \frac{1-\alpha^{\ell}}{1-\alpha} = \frac{\alpha^{\ell}}{1-\alpha}$$

Making the appropriate substitutions in the expression for $R_y(\ell)$, we have:

$$R_y(\ell) = \frac{1}{(\alpha_a-\alpha_b)^2}[\alpha_a^{2-\ell}(\frac{\alpha_a^{2\ell}}{1-\alpha_a^2}) - \alpha_a\alpha_b^{1-\ell}(\frac{\alpha_a^{\ell}\alpha_b^{\ell}}{1-\alpha_a\alpha_b}) - \alpha_b\alpha_a^{1-\ell}(\frac{\alpha_a^{\ell}\alpha_b^{\ell}}{1-\alpha_a\alpha_b}) + \alpha_b^{2-\ell}(\frac{\alpha_b^{2\ell}}{1-\alpha_b^2}) \quad (\ell \geq 0) \tag{F.4}$$

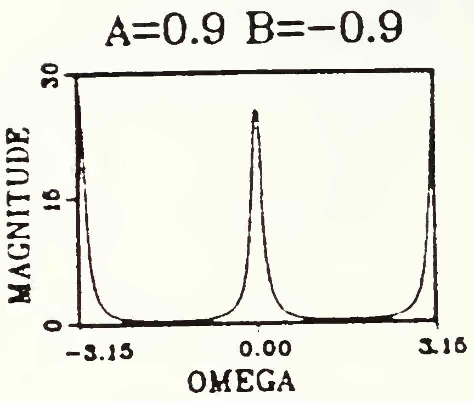
Combining terms yields the final expression:

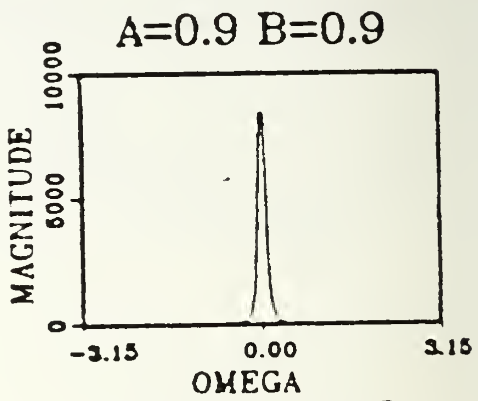
$$R_y(\ell) = \frac{1}{(\alpha_a-\alpha_b)^2}[\frac{\alpha_a^{2+\ell}}{1-\alpha_a^2} - \frac{\alpha_a^{\ell+1}\alpha_b+\alpha_a\alpha_b^{\ell+1}}{1-\alpha_a\alpha_b} + \frac{\alpha_b^{2+\ell}}{1-\alpha_b^2}] \tag{F.5}$$

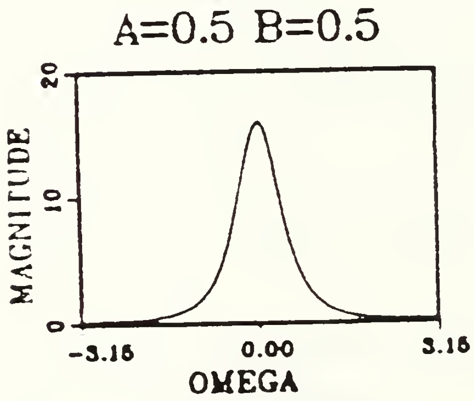
APPENDIX G

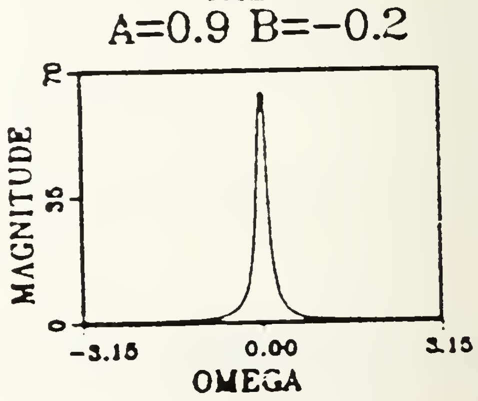
GRAPHICAL RESULTS FOR THE POWER SPECTRUM AND AUTO-CORRELATION FUNCTION FOR THE FOUR POLE AUTO-REGRESSIVE MODEL (WITH TWO POLES ON THE REAL AXIS)

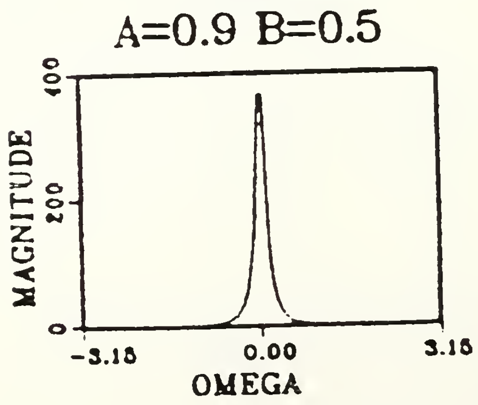
Power Spectrum

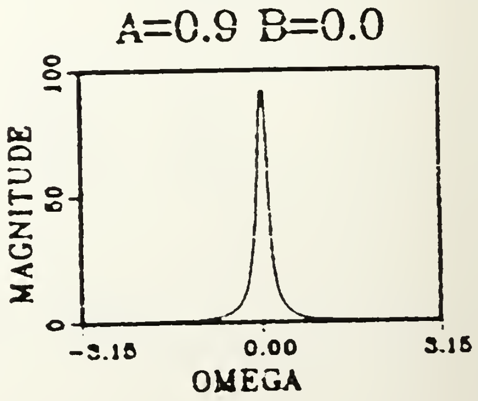


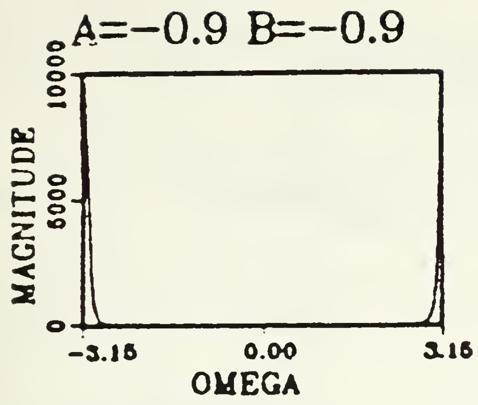


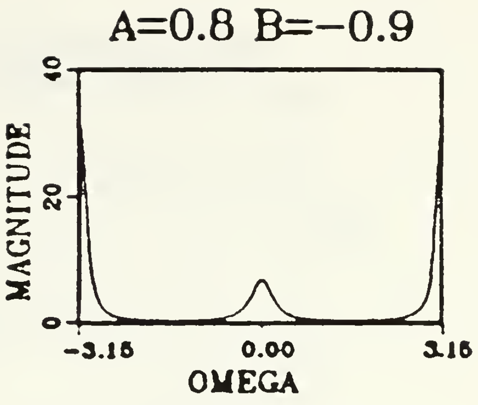


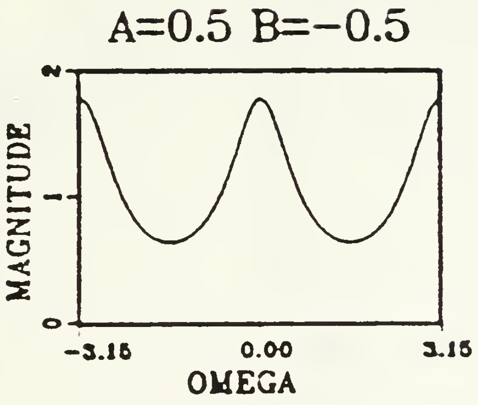




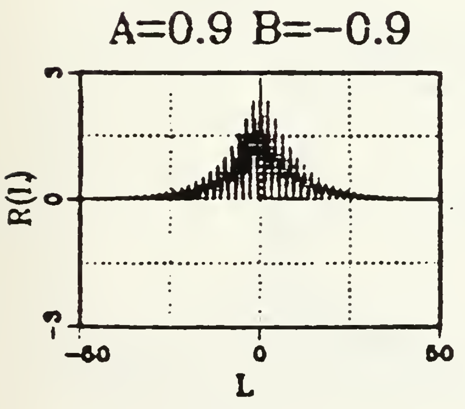


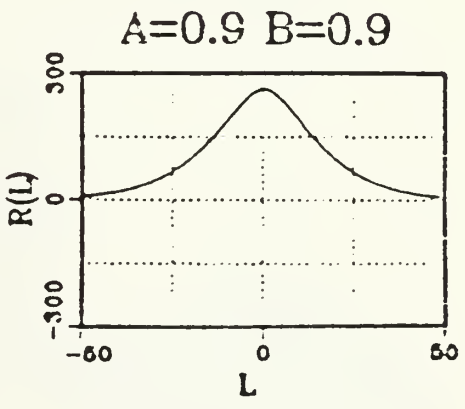


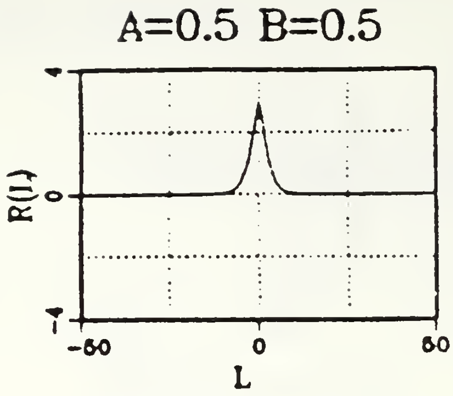


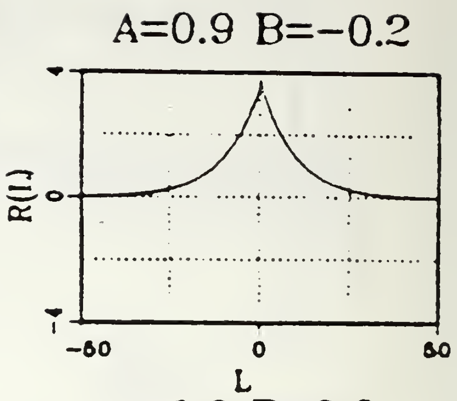


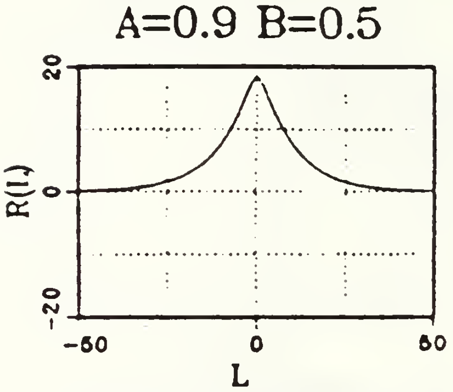
Autocorrelation Function

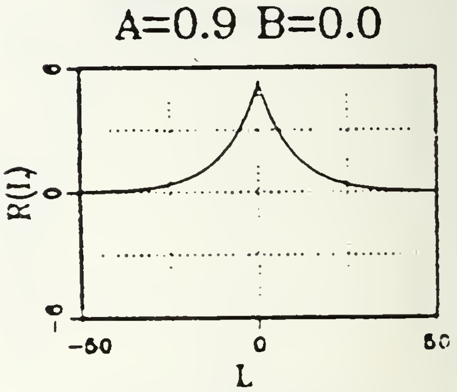


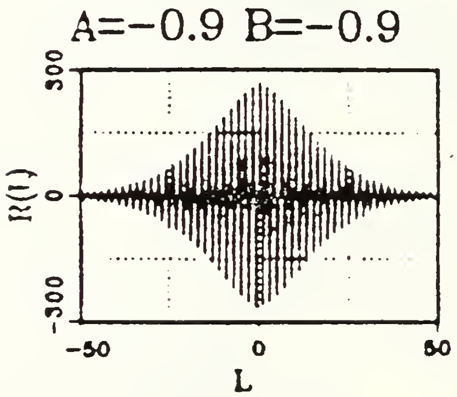


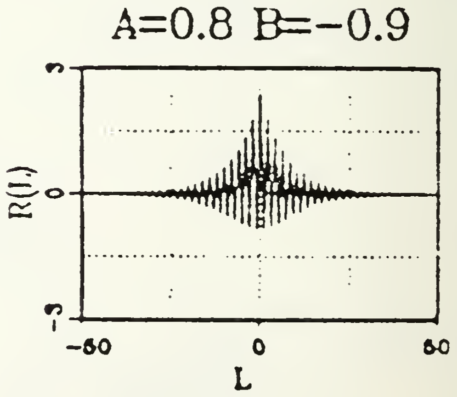

A=0.5 B=0.5
R(L)
4
0
-4
-50
0
50
L

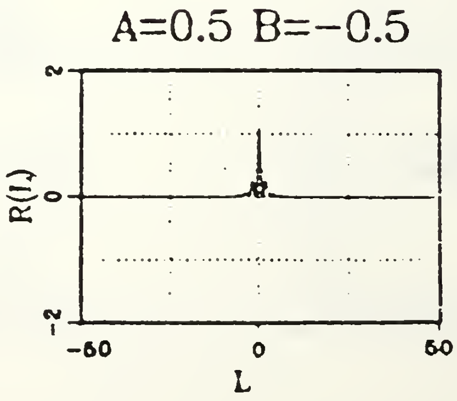

A=0.9 B=-0.2
R(L)
4
0
-4
-50
0
50
L


A=0.9 B=0.5
R(L)
20
0
-20
-50
0
50
L


A=0.9 B=0.0
R(L)
6
0
-6
-50
0
50
L


A=-0.9 B=-0.9
R(L)
300
0
-300
-50
0
50
L

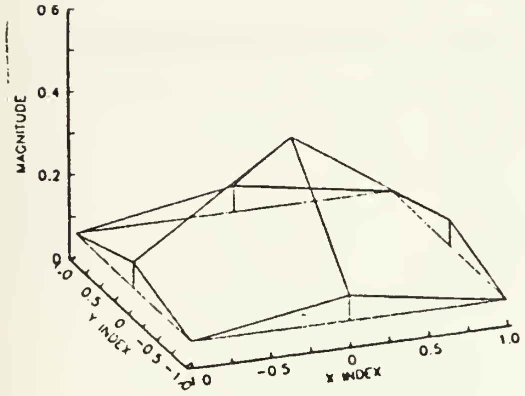

A=0.8 B=-0.9
R(L)
3
0
-3
-50
0
50
L


A=0.5 B=-0.5
R(L)
2
0
-2
-50
0
50
L

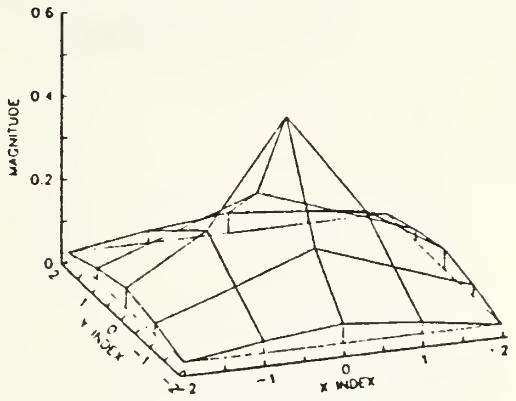
APPENDIX H

LAPLACIAN INVERSE FILTER FORMS

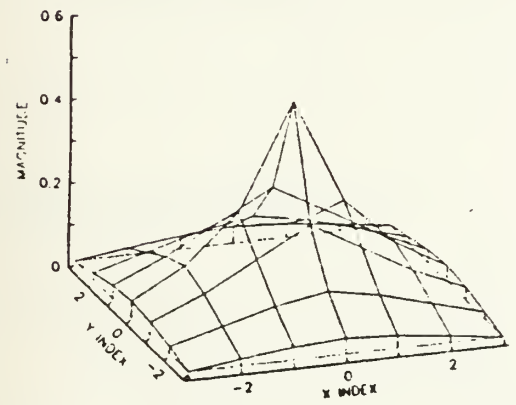
INVERSE FIR FILTER
3×3 UNNORMALIZED



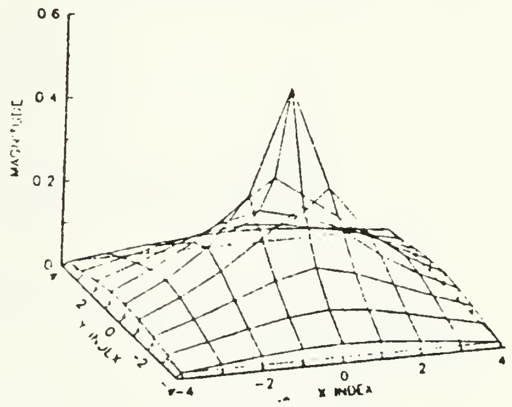
INVERSE FIR FILTER
5×5 UNNORMALIZED



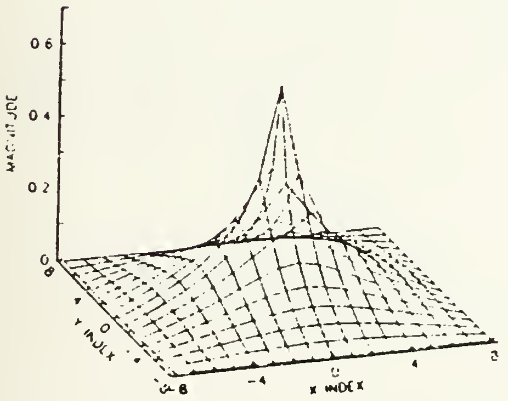
INVERSE FIR FILTER
7×7 UNNORMALIZED



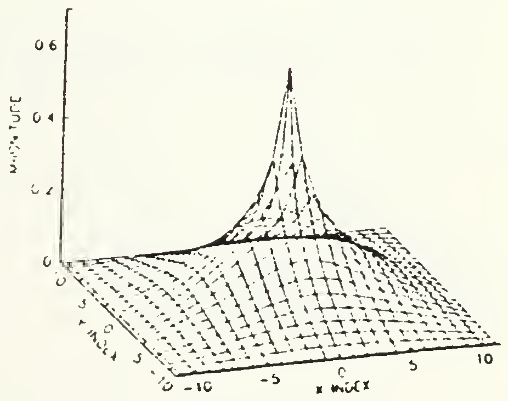
INVERSE FIR FILTER
9×9 UNNORMALIZED



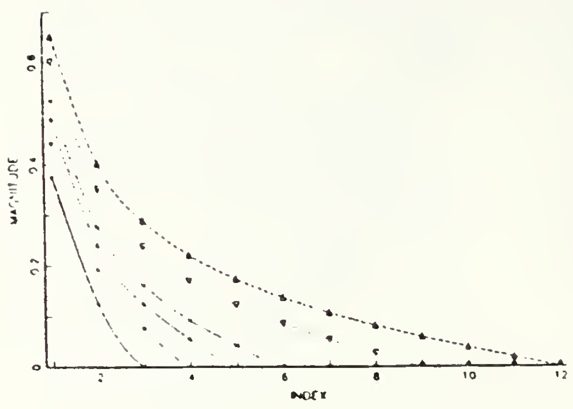
INVERSE FIR FILTER
15×15 UNNORMALIZED



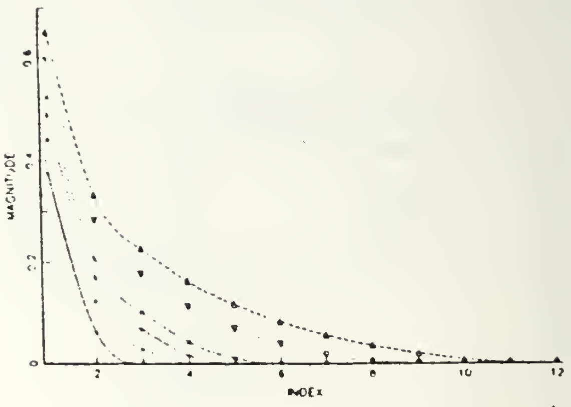
INVERSE FIR FILTER
21×21 UNNORMALIZED



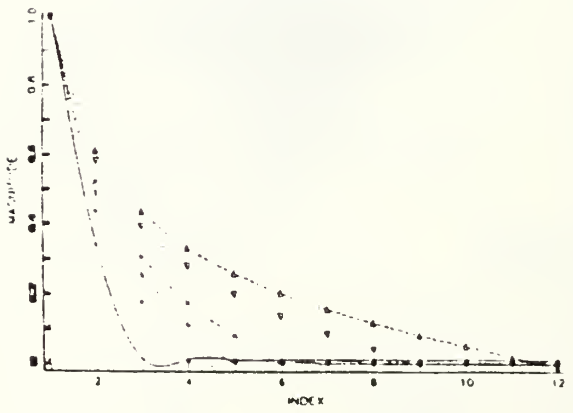
LAPLACIAN INVERSE FILTERS (3×3 TO 21×21)
HORIZONTAL CROSS SECTION



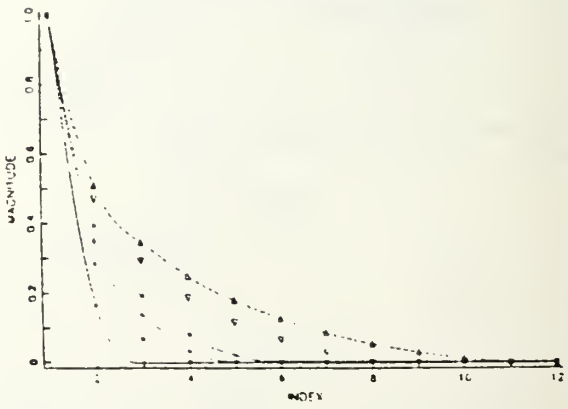
LAPLACIAN INVERSE FILTERS (3×3 TO 21×21)
DIAGONAL CROSS SECTION



LAPLACIAN INVERSE FILTERS (3×3 TO 21×21)
HORIZONTAL CROSS SECTION (NORMALIZED)



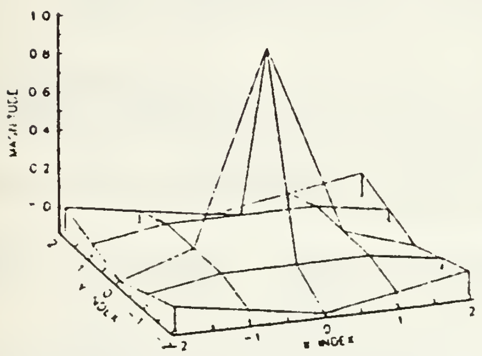
LAPLACIAN INVERSE FILTERS (3×3 TO 21×21)
DIAGONAL CROSS SECTION (NORMALIZED)



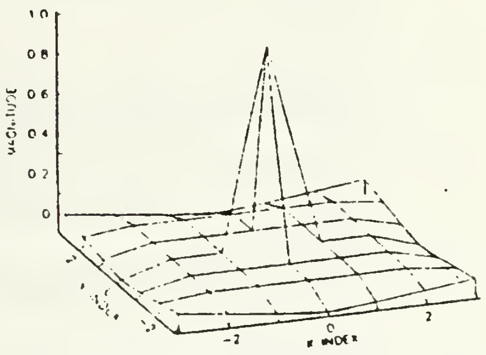
APPENDIX I

CONVOLUTION OF LAPALACIAN DIFFERENCE OPERATOR AND VARIOUS SIZE FIR INVERSE FILTERS

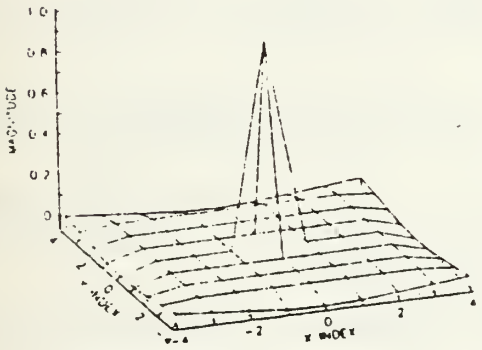
CONVOLUTION OF LAPLACIAN AND ITS 3×3 INVERSE



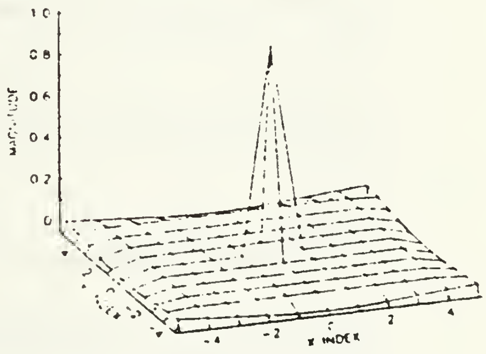
CONVOLUTION OF LAPLACIAN AND ITS 5×5 INVERSE



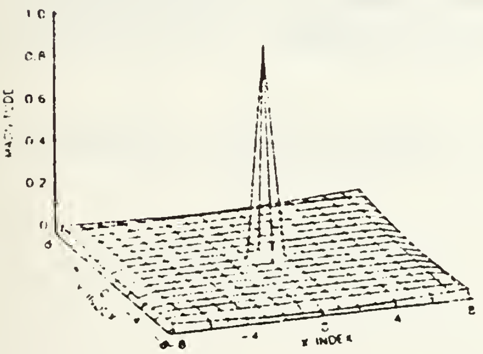
CONVOLUTION OF LAPLACIAN AND ITS 7×7 INVERSE



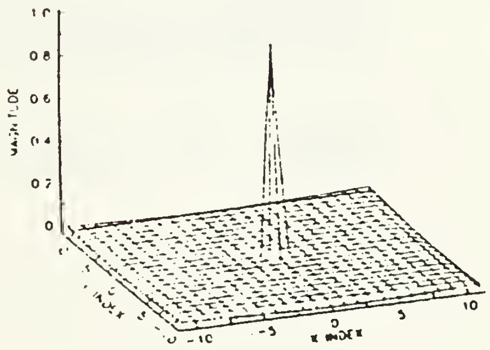
CONVOLUTION OF LAPLACIAN AND ITS 9×9 INVERSE



CONVOLUTION OF LAPLACIAN AND ITS 15×15 INVERSE



CONVOLUTION OF LAPLACIAN AND ITS 21×21 INVERSE



LIST OF REFERENCES

1. Box, G.E.P., and Jenkins, G.M., Time Series Analysis, Forecasting and Control, Holden-Day, 1976.

2. Therrien, C.W., "Relations Between 2-D and Multichannel Linear Prediction," IEEE Transactions on Acoustics, Speech, and Signal Processing, Vol. ASSP-29, No. 3, June 1981.

3. Dudgeon, D.E., and Mersereau, R.M., Multidimensional Digital Signal Processing, Prentice-hall, Inc., 1984.

4. Orfanidis, S.J., Optimum Signal Processing: An Introduction, Macmillan Publishing Co., 1985.

5. Oppenheim, A.V. and Schafer, R.W., Digital Signal Processing, Prentice Hall, Inc., 1975.

6. Peebles, P.Z. Jr., Communications System Principles, Addison-Wesley Publishing Company, Inc., 1976.

7. Cadzoq, J.A., Discrete Time Systems, Prentice-Hall, Inc., 1973.

8. Gonzalez, R.C. and Wintz, P., Digital Image Processing, Addison-Wesley Publishing Company, Inc., 1977.

9. Weber, A.G., "USC SIPI Report 101, Image Data Base," Signal and Image Processing Institute, University of Southern California, February 1986.

10. Brodatz, P., Textures: A Photographic Album for Artists and Designers, Dover, 1966.

11. Modestino, J.W. and Fries, R.W., "Construction and Properties of a Useful Two-Dimensional Random Field," IEEE Transactions, Information Theory, Vol. IT-26, No. 1, January 1980.

12. Beyer, W.H., CRC Standard Mathematical Tables, CRC Press, Inc., 1984.

INITIAL DISTRIBUTION LIST

		No. Copies
1.	Defense Technical Information Center Cameron Station Alexandria, Virginia 22304-6145	2
2.	Library, Code 0142 Naval Postgraduate School Monterey, California 93943-5002	2
3.	Department Chairman, Code 62 Electrical and Computer Engineering Dept. Naval Postgraduate School Monterey, California 93943-5000	1
4.	Professor Charles W. Therrien, Code 62Ti Electrical and Computer Engineering Dept. Naval Postgraduate School Monterey, California 99343-5000	3
5.	Professor Roberto Cristi, Code 62Cx Electrical and Computer Engineering Dept. Naval Postgraduate School Monterey, California 93943-5000	3
6.	LT Steven C. Rathmanner 1453 N. 52nd Ave. Pensacola, Florida 32506	3
7.	Director, Research Administration, Code 012 Naval Postgraduate School Monterey, California 93943-5000	1
8.	Hamdy El-Shaer SMC #1616 Naval Postgraduate School Monterey, California 93943-5000	1

Thesis
R24345 Rathmanner
c.1 Image texture generation using autoregressive integrated moving average (ARIMA) models.

Thesis
R24345 Rathmanner
c.1 Image texture generation using autoregressive integrated moving average (ARIMA) models.